



LarKC

*The Large Knowledge Collider:
a platform for large scale integrated reasoning and Web-search*

FP7 – 215535

D6.6 – 2nd periodic report on data and performances

Coordinator: Daniele Dell'Aglio (Cefriel)

With contributions from: Irene Celino, Emanuele Della Valle, Kono Kim, Stanley Park, Florian Steinke

Quality Assessor: Barry Bishop (STI-Innsbruck)

Quality Controller: Emanuele Della Valle (Cefriel)

| | |
|----------------------|--------------------------|
| Document Identifier: | LarKC/2008/D6.6/V1.0 |
| Class Deliverable: | LarKC EU-IST-2008-215535 |
| Version: | Version 1.0 |
| Date: | September 30th, 2009 |
| State: | Final |
| Distribution: | Public |



EXECUTIVE SUMMARY

This document updates D6.4 – “1st periodic report on data and performances” with a description of additional data sources we analyzed in order to consider them in the development of our activities and our scenario.

We also start to test the Urban LarKC workflows we prepared in previous months in order to obtain some indications about the performance of the applications we are developing over the platform. Considering the comments received by the reviewer, we consider a fewer number of measurements than the ones described in D6.2 – “Templates of periodic report on data and performances”. The document contains a description of the methodologies that we followed and that we’d like to use for the next deliverables and the results that we obtained in these first initial tests.



DOCUMENT INFORMATION

| | | | |
|---------------------------|--|----------------|-------|
| IST Project Number | FP7 – 215535 | Acronym | LarKC |
| Full Title | The Large Knowledge Collider: a platform for large scale integrated reasoning and Web-search | | |
| Project URL | http://www.larkc.eu/ | | |
| Document URL | | | |
| EU Project Officer | Stefano Bertolo | | |

| | | | | |
|---------------------|---------------|-----|--------------|--|
| Deliverable | Number | 6.6 | Title | 2 nd periodic report on data and performances |
| Work Package | Number | 6 | Title | Urban Computing |












| | | | | |
|----------------------------|--|-----|---------------|-----|
| Date of Delivery | Contractual | M18 | Actual | M18 |
| Status | Draft | | final ■ | |
| Nature | prototype <input type="checkbox"/> report ■ dissemination <input type="checkbox"/> | | | |
| Dissemination level | public ■ consortium <input type="checkbox"/> | | | |

| | | | | |
|---------------------------|---|--------------------|---------------|------------------------------|
| Authors (Partner) | Daniele Dell’Aglia, Irene Celino, Emanuele Della Valle (Cefriel), Kono Kim, Stanley Park (Saltlux), Florian Steinke (Siemens) | | | |
| Responsible Author | Name | Daniele Dell’Aglia | E-mail | daniele.dellaglio@cefriel.it |
| | Partner | Cefriel | Phone | +39 (02) 23954-324 |

| | |
|-------------------------------------|--|
| Abstract (for dissemination) | This document is the second step of reporting about the collection of data sources and the evaluation results of the performance of the early Urban Computing demonstrators. |
| Keywords | data sets, measurements, tests, performances, stress tests, evaluation, periodic report, use case, urban computing |

| Version Log | | | |
|--------------------|-----------------|---------------|---|
| Issue Date | Rev. No. | Author | Change |
| August 3, 2009 | 1 | Daniele | First draft of the document |
| August 5, 2009 | 2 | Kono | Input for the reviewer comments response |
| August 29, 2009 | 3 | Florian | Addition in data reports Traffic prognosis Input for the reviewer comments response |
| September 4, 2009 | 4 | Daniele | Addition of stress tests |
| September 4, 2009 | 5 | Emanuele | Addition in data reports |
| September 8, 2009 | 6 | Daniele | Addition of LarKC performance tests |
| September 9, 2009 | 7 | Irene | Response for reviewer comments |
| September 11, 2009 | 8 | Daniele | Addition of Urban Computing workflows related tests |
| September 14, 2009 | 9 | Daniele | Introduction, conclusions |
| September 16, 2009 | 10 | Kono | OntoBroker tests |
| September 29, 2009 | 11 | Daniele, all | Corrections on all the quality assurer remarks and finalization of the document |

PROJECT CONSORTIUM INFORMATION

| Participant's name | Partner | Contact |
|--|--|--|
| Semantic Technology Institute Innsbruck, University of Innsbruck |   | Prof. Dr. Dieter Fensel, Semantic Technology Institute (STI), universitaet Innsbruck, Innsbruck, Austria, E-mail: dieter.fensel@sti-innsbruck.at |
| AstraZeneca AB |  | Bosse Andersson AstraZeneca Lund, Sweden Email: bo.h.andersson@astrazeneca.com |
| CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA |  | Emanuele Della Valle, CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA, Milano, Italy, Email: emanuele.dellavalle@cefriel.it |
| CYCROP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O. |  | Michael Witbrock, CYCORP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O., Ljubljana, Slovenia, Email: witbrock@cyc.com |
| Höchstleistungsrechenzentrum, Universitaet Stuttgart |  | Georgina Gallizo, Höchstleistungsrechenzentrum, Universitaet Stuttgart, Stuttgart, Germany, Email: gallizo@hlrs.de |
| MAX-PLANCK GESELLSCHAFT ZUR FOERDERUNG DER WISSENSCHAFTEN E.V. |  | Dr. Lael Schooler Max-Planck-Institut für Bildungsforschung Berlin, Germany Email: schooler@mpib-berlin.mpg.de |
| Ontotext Lab, Sirma Group Corp |  | Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: atanas.kiryakov@sirma.bg |
| SALTLUX INC. |  | Tony Lee, SALTLUX INC, Seoul, Korea, Email: tony@saltlux.com |
| SIEMENS AKTIENGESELLSCHAFT |  | Dr. Volker Tresp, SIEMENS AKTIENGESELLSCHAFT, Muenchen, Germany, E-mail: volker.tresp@siemens.com |
| THE UNIVERSITY OF SHEFFIELD |  | Prof. Dr. Hamish Cunningham, THE UNIVERSITY OF SHEFFIELD Sheffield, UK, Email: h.cunningham@dcs.shef.ac.uk |



| | | |
|--|---|---|
| <p>VRIJE UNIVERSITEIT AMSTERDAM</p> |  | <p>Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM, Amsterdam, Netherlands, Email: Frank.van.Harmelen@cs.vu.nl</p> |
| <p>THE INTERNATIONAL WIC INSTITUTE, BEIJING UNIVERSITY OF TECHNOLOGY</p> |  | <p>Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE, Mabeshi, Japan, Email: zhong@maebashi-it.ac.jp</p> |
| <p>INTERNATIONAL AGENCY FOR RESEARCH ON CANCER</p> |  | <p>Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RESEARCH ON CANCER, Lyon, France, Email: brennan@iarc.fr</p> |
| <p>INFORMATION RETRIEVAL FACILITY</p> |  | <p>John Tait, INFORMATION RETRIEVAL FACILITY Vienna, Austria Email : john.tait@ir-facility.org</p> |



TABLE OF CONTENTS

| | |
|---|-----------|
| 1. INTRODUCTION | 7 |
| 2. ADDRESSING REVIEWERS COMMENTS..... | 8 |
| 2.1. PERFORMANCE EVALUATION ISSUE | 8 |
| 2.2. LARGE DATA ISSUE | 8 |
| 3. PERIODIC REPORT ON DATA | 10 |
| 3.1. TRAFFIC SENSOR DATA SOURCE..... | 10 |
| 3.2. ILMETEO.IT HISTORICAL WEATHER DATA..... | 11 |
| 3.3. GLUE SOCIAL NETWORK INTERACTIONS STREAMING DATA SOURCE..... | 12 |
| 4. PERIODIC REPORT ON PERFORMANCE | 13 |
| 4.1. OPERATIONAL RESEARCH PATH FINDING PERFORMANCE TEST – WORKFLOW 1A | 14 |
| 4.2. OPERATIONAL RESEARCH PATH FINDING PERFORMANCE TEST – WORKFLOW2A..... | 17 |
| 4.3. OPERATIONAL RESEARCH PATH FINDING PERFORMANCE TEST – WORKFLOW 2B | 19 |
| 4.4. RULE-BASED PATH FINDING PERFORMANCE TEST | 21 |
| 4.5. ALPHA URBAN LARKC STRESS TEST – PATH FINDING WORKFLOW | 24 |
| 4.6. ALPHA URBAN LARKC STRESS TEST – MONUMENT WORKFLOW | 27 |
| 4.7. ALPHA URBAN LARKC STRESS TEST – EVENT WORKFLOW | 31 |
| 4.8. RULE-BASED PATH FINDING STRESS TEST | 35 |
| 4.9. ALPHA URBAN LARKC – MONUMENT WORKFLOW – DATA QUALITY TEST | 36 |
| 4.10. ALPHA URBAN LARKC – EVENT WORKFLOW – DATA QUALITY TEST..... | 37 |
| 4.11. TRAFFIC PROGNOSIS | 38 |
| 5. LESSON LEARNED AND CHALLENGES FOR LARKC..... | 40 |
| 6. CONCLUSIONS..... | 42 |
| 7. REFERENCES | 42 |



1. Introduction

This deliverable represents the second periodic report on data and performance of the Urban Computing use case in LarKC. It is based on the templates provided in D6.2 “Templates of periodic report on data and performances” and it follows the early results presented in D6.4 “1st periodic report on data and performances”. With that respect, this report includes the newly acquired data sources and it selects a set of possible measurements from those envisioned in D6.4 that we used to perform tests on the existing Urban Computing demonstrators developed in WP6. In this deliverable we do not enter in details of those demonstrators, since they are included in the companion deliverable D6.5 “Urban Computing environment v1”.

The objective of this deliverable, as its title suggests, is twofold.

On the one hand, this report is aimed at providing an update on the list of data sources related to the Urban Computing use case; since the whole project aims at providing solutions to handle large scale of data, among other sources in this report we describe a very large archive of Milano traffic sensor recordings over several years. Those data sources will be useful to test LarKC ability to tackle the scalability issue.

On the other hand, this deliverable reports on the first tests we performed on the early Urban Computing demonstrators built on the available LarKC platform release and developed within WP6. Those tests are useful to understand possible bottlenecks, to suggest possible improvements, to derive scientific or technological challenges to propose to other LarKC technical work-packages. They will also serve as comparison for future improvements of both the Urban Computing applications and the platform; since we expect both to improve, the results reported in this document will be compared to the homogeneous results of future software releases to identify advancements.

The deliverable is structured as follows. Chapter 2 address in more details the comments expressed by reviewers in the first project period report after the first LarKC review in June 2009. Chapter 3 describes the new data sources as per the template used also in the previous deliverable D6.4. Chapter 4 presents our tests in terms of the adopted methodology, the tested demonstrators, the results of this testing and the interpretation we can give to those results. On the basis of such experience, Chapter 5 reports some lesson learned and possible challenges to be solved by LarKC in the context of Urban Computing. Finally, some conclusions are offered in Chapter 6.



2. Addressing reviewers comments

In the report of the first review of the LarKC project, the reviewers included some comments and recommendations as feedbacks to the previous WP6 deliverables. In this section, we would like to provide a broader explanation about how we mean to address their remarks.

2.1. Performance evaluation issue

In commenting the acceptance of deliverable D6.2 - “Templates of periodic report on data and performances”, the reviewers wrote the following remarks:

“The deliverable considers the factors that will be relevant for characterizing data sources and for evaluating the performance of (the urban computing) application(s). Careful evaluation will be critical for measuring the success or otherwise of the project. This report makes a useful start in establishing a suitable framework, but it will no doubt need to be refined as the project progresses. One problem is that very many measures are considered. It will be useful to fix on a smaller number of critical measurements that can be used to compare the performance of LarKC with that of other systems and technologies.”

WP6 partners raised the same issue during the writing of the deliverable. However, we concluded that identifying new measures would have cost more than identifying them in advance and selecting them later when measuring. Moreover, those numerous measures can be explained partly because LarKC uses many technologies such as information retrieval, reasoning, natural language processing, and machine learning. We categorized performance measures under different categories like Scale, Heterogeneity, Reusability, and Statistical Measurement. At the beginning, we will focus on measuring *Scale* factors and then we will continue with *Heterogeneity* measurements, since Scale is the main purpose of LarKC “web-scale reasoning”, while coping with heterogeneity is an important requirement for LarKC. For the time being, *Reusability* and *Statistical Measurement* will have low priority; *Reusability* will measure the number and kind of functionalities provided by plug-ins and *Statistical Measurement* will measure the performance of machine learning modules in LarKC.

In this deliverable, therefore, we apply only a subset of all possible measurements defined in D6.2. In particular, we define three kinds of test to evaluate the current achievements of LarKC in the Urban Computing:

- 1) *Measurement of the "overhead" introduced by the LarKC platform*: the Urban Computing demonstrators developed within WP6 in the first project period have a respective "term of comparison" (cf. D6.3, Section 2), which provides the same functionality, but it is based on "standard" technologies; those terms of comparison were realized in a first project stage, when the LarKC platform was not ready yet, and completed in the last few months. The purpose of this evaluation is to understand how much the flexibility and extensibility of the platform impacts its efficiency in terms of performance and response time when satisfying the need of our Urban Computing scenarios.
- 2) *Stress tests*: this is aimed at measuring the response time with multiple concurrent users and similar factors. The purpose of this evaluation is to test the stability and "resistance" of Urban LarKC workflows to computation loads.
- 3) *Quality evaluation of the data results*: this is a scenario-specific evaluation which aims at measuring the quality of Urban LarKC query results. The purpose of this evaluation is to evaluate the strategies to get to the results and provide further requirements for technical WPs to develop better plug-ins.

In the next evaluation deliverables, we will perform those tests again, in order to show any improvements in LarKC performance; if necessary, we will extend the set of measurements accordingly to the new requirements and achievements that can come out as the project proceeds.

2.2. Large data issue

In commenting the acceptance of deliverable D6.4 “First periodic report on data and performances”, the reviewers wrote the following remark:

“At this stage of the project only information on data sources is provided. Very large scale data sources still seem to be lacking.”



In the time between the first project review and the delivery of this document, we acquired a new data source, consisting of historical traffic sensor data from Milano Municipality (see also Section 3.1) whose scale is in the order of 10^9 tuples, that would mean around 10^{10} statements if expressed in RDF. The next step is to use this data for predicting traffic conditions using machine learning techniques (cf.6.5, Section 4). For those purposes, it is not strictly necessary to convert the aforementioned relational data into an RDF format.

However, the conversion of this data into RDF and its efficient querying could nevertheless be an interesting technological challenge for the LarKC project for several reasons.

First of all, the sheer size of the data makes a conversion into RDF problematic. Storing everything in one file, the file size would most likely exceed several hundred GB, since in a densely packed binary database format the data is already about 1GB.

Secondly, the representational form in RDF is an interesting question: the data consists mainly of massive time series together with some side information on where and how the data was acquired. Therefore, it would be interesting to develop a standard concept for managing time stamps (named graphs? explicit statements?); additionally, due to the sequential nature of the data, it seems like an unnecessary and wasteful overhead to store joint information for each datum separately, (e.g. from which sensor id it was acquired). Instead, it would be reasonable to think about forms to represent whole blocks of similar data jointly (something close to the concept of RDF Molecules), while still allowing simple querying access.

As mentioned elsewhere, we want to pose these questions as a challenge to the whole LarKC consortium; in particular, WP4 could contribute some solutions here. This one and other challenges that came to our minds during the last months are summarized and offered to other WPs in Chapter 5.

Apart from traffic data, we also collected other large-scale datasets, which are described within Chapter 1 of this deliverable. In addition, WP6 partners are evaluating the possibility of re-using the results of a recent initiative external to LarKC named “*LinkedGeoData*”¹. As the respective home page explains “*LinkedGeoData* is an effort to add a spatial dimension to the Web of Data / Semantic Web. *LinkedGeoData* uses the information collected by the *OpenStreetMap* project² and makes it available as an RDF knowledge base according to the Linked Data principles. It interlinks this data with other knowledge bases in the Linking Open Data initiative.” From that initiative, WP6 could obtain the RDF dumps of the data or could remotely access the exposed services to query the data. This could be a good starting point to study the extension of our work related to the Milano area to a larger scale. This will also challenge the LarKC platform to work with a very large amount data and to cope with harder scalability issues.

Finally, we would like to note that the so-called alpha Urban LarKC demonstrator already involved large-scale data sets. In fact, in order to retrieve general information about Milano, the application interacts with a part of the LOD cloud (namely DBpedia), by means of the references provided by Sindice. In the following, we could also try to access the LOD cloud with other less “middlemen” approaches, for example by using the Linked Data Semantic Repository (LDSR) developed by LarKC project partners. This could also be useful to compare the different approaches to retrieve information from the already-in-place Semantic Web.

¹ See <http://linkedgeodata.org/> for more details on this initiative.

² See <http://www.openstreetmap.org/> to learn more.



3. Periodic report on Data

3.1. Traffic sensor data source

| | | | |
|---|--|--|---|
| Data Source: Traffic Sensor Data from Milano | | | |
| Report ID | | | |
| Section 1 | | Data source metadata | |
| Name | | Traffic Sensor Data from Milano | |
| Producer/Owner | | Agenzia Mobilità e Ambiente Srl (http://www.ama-mi.it) | |
| Description | | <p>Relational database with information about speed, flow, and occupation rate from stationary sensors on roads of Milano. Some sensors also classify vehicles into different classes.</p> <p>Additional Tables containing</p> <ul style="list-style-type: none"> • information whether a certain day was a holiday and whether it was in a low traffic season, • descriptions of the different vehicle classes, • descriptions of the different time slots (5min intervals). <p>Detailed documentation of all table columns is available together with the data.</p> | |
| Namespace/Web Address | | Not applicable | |
| Availability | | Obtained with an agreement with the Milano Municipality | |
| Download/Upload/Acquisition date | | July 2009 | |
| Version | | 1 (July 2009) | |
| Physical size | | 900 MB (packed database dump), 25GB (unpacked database) | |
| Nature of data type | | Time series | |
| Quality of the data source | | Average | |
| Section 2 | | “semantics” of the data source | |
| Typology of data | | Relational Database | |
| Geographic coverage of data | | Milano area | |
| Applied systems | | PostgreSQL Database | |
| Existence of schema/ontology | | See remarks | |
| Existing links with other data-sources | | Specific data structures allow to link sensors with Milano road map | |
| Possible linkage to other data-sources | | Other geographic datasets | |
| Scale of data | | The time-series tables contain about 10^9 rows with either 5 or 7 columns. | |
| Section 3 | | Data source format | |
| Format of data | | PostgreSQL Database Dump | |
| Generation method | | Unknown | |
| Support query language | | See remarks | Total no. of statements About 10^{10} |
| Support triple type | | See remarks | |
| No. of explicit statements | | If converted to RDF, about 10^{10} | |
| Noise, Uncertainty and inconsistency of data | | <ul style="list-style-type: none"> • Not all sensors are available for all time steps. For some sensors more than 90% of possible time slots are not measured. • Sometimes sensors are unreliable (e.g. speed measurements at night often attain zero or maximal values). Reliability scores are given for the non-classifying sensors. • There are also some (possibly storage related) outliers in the flow measurements. | |
| Remarks | | | |
| | | The time series data has not been converted to RDF so far since doing this in a straightforward manner would result in too large files. The data is mostly in the form of time series. Thus, it would be an interesting challenge to the other work packages of LarKC to develop ideas and | |



| | |
|--|---|
| | methods to efficiently store and retrieve large time series in a semantic web format. |
|--|---|

The way we will make use of this data set is described in deliverable D6.5, Section 4 and the respective evaluation is sketched in Section 4.11 of this document.

3.2. *ilMeteo.it Historical Weather Data*

| | | | |
|---|--|---------------------------------------|---------------------------------------|
| Data Source: ilMeteo.it | | | |
| Report ID | | | |
| Section 1 | | Data source metadata | |
| Name | ilMeteo.it archive of weather data in Italy | | |
| Producer/Owner | ilMeteo s.r.l. (http://www.ilmeteo.it) | | |
| Description | ilMeteo.it is a portal devoted to provide weather data for all Italian municipalities. ilMeteo.it offers weather prediction every day, but it also offer historical data of the weather measurements of past years. The archive gives the recordings from 1973 to today. | | |
| Namespace/Web Address | http://www.ilmeteo.it/portale/archivio-meteo/ | | |
| Availability | Free to download | | |
| Download/Upload/Acquisition date | Continuous, since the archives are updated daily | | |
| Version | 1.0 | | |
| Physical size | N/A | | |
| Nature of data type | Weather data | | |
| Quality of the data source | Good | | |
| Section 2 | | “semantics” of the data source | |
| Typology of data | Information about temperature, rain, humidity, wind and other information, per each day, per each Italian municipality (there are more than 8.000 municipalities). | | |
| Geographic coverage of data | Italy | | |
| Applied systems | Weather predictions | | |
| Existence of schema/ontology | Not available; however, the content can be represented using a simple schema | | |
| Existing links with other data-sources | No | | |
| Possible linkage to other data-sources | Geographic data sources | | |
| Scale of data | Order of magnitude: 10^8 records | | |
| Section 3 | | Data source format | |
| Format of data | Comma separated values file (CSV) | | |
| Generation method | Unknown | | |
| Support query language | See remarks | Total no. of statements | Potentially, around 10^9 statements |
| Support triple type | See remarks | | |
| No. of explicit statements | See remarks | | |
| Noise, Uncertainty and inconsistency of data | Some values with low reliability are explicitly marked. | | |
| Remarks | | | |
| | The native data is not in RDF format. We can perform a batch transformation to obtain RDF data and then to query via SPARQL. | | |

The weather data is very useful for the traffic prediction, since their integration with the traffic data can give useful insight on the causes of traffic congestions. See also deliverable D6.5, Chapter 4 to learn more details about the joint use of traffic and weather data.



3.3. Glue social network interactions streaming data source

| | | | |
|--|--|---|-------------------|
| Data Source: Glue social network interactions | | | |
| Report ID | | | |
| Section 1 | | Data source metadata | |
| Name | | Glue social network interactions | |
| Producer/Owner | | AdaptiveBlue, Inc. (http://getglue.com/) | |
| Description | | Glue enables users to connect with their friends on the Web based on the pages the users visit online. Using semantic recognition technologies to automatically identify books, music, movies, wines, stocks, movie stars and many other similar topics, it generates a continuous stream of the identified objects which is accessible in real time using a REST API. The REST request http://api.getglue.com/v1/glue/recent returns the 250 most recent public interactions. We adopted the GRDDL approach and implemented a simple way to translate the resulting XML into a RDF stream. | |
| Namespace/Web Address | | http://c-sparql.cefriel.it/sdow-demo/RDFstream.html | |
| Availability | | See Terms Of Services at http://getglue.com/api | |
| Download/Upload/Acquisition date | | Continuous, measuring it in triple per second (t/s), it has a rate of 1 t/s | |
| Version | | 1.0 | |
| Physical size | | Not applicable; between 3.8.2009 and 4.9.2009 we generated a RDF stream whose size, if stored, would have been of 2.768.434 triples. | |
| Nature of data type | | Streaming data | |
| Quality of the data source | | Good | |
| Section 2 | | “semantics” of the data source | |
| Typology of data | | Interaction of people subscribed in Glue Social Network with Web resources (books, music, movies, wines, stocks, movie stars and many other similar topics) | |
| Geographic coverage of data | | Worldwide | |
| Applied systems | | Social Network, Recommendation Systems | |
| Existence of schema/ontology | | The content can be represented using: Semantically-Interlinked Online Communities (SIOC), Dublin Core and RDFS | |
| Existing links with other data-sources | | No | |
| Possible linkage to other data-sources | | DBpedia, DBtunes, BestBuy and other data sources in the LOD cloud | |
| Scale of data | | See physical size | |
| Section 3 | | Data source format | |
| Format of data | | Custom XML schema described at http://getglue.com/api | |
| Generation method | | Generated by the Glue Firefox plug-in installed by the users | |
| Support query language | | Total no. of statements | See physical size |
| Support triple type | | See remarks | |
| No. of explicit statements | | See physical size | |
| Noise, Uncertainty and inconsistency of data | | Not assessed so far | |
| Remarks | | | |
| The native data is not in an RDF format. We can perform a GRDDL transformation to obtain RDF data. | | | |

We will integrate this data source the same way we integrated other general-purpose sources like DBpedia and Eventful. For example, whenever a Glue user happens to visit a Wikipedia page about Milano, this event will be recorded in the Glue data stream and can therefore be exploited by our Urban Computing applications. Beside the natural relationship between Wikipedia and DBpedia (which is a natural link between the two sources), Glue can be used also to come to know other users of the Glue social network which are interested in the same topic (“friends” and “neighbours” in Glue terminology). Finally, Glue is available also as an iPhone app and this data source can help in taking a further step towards the development of mobile Urban Computing applications.



4. Periodic report on Performance

In order to evaluate the LarKC platform in the Urban Computing scenario we scheduled several kinds of tests:

- *Performance comparison tests*: the main goal is to collect information related to LarKC, such as estimations of overhead (in terms of execution time) that are introduced by the platform in the Urban Computing applications. In this test we used the following workflows:
 - o workflow 1a: presented in D6.3, Section 2.5.4, it loads an RDF file containing the graph or a subgraph of Milano (the test is described in Section 4.1);
 - o workflow 2a: presented in D6.3, Section 2.5.4, it retrieves a subgraph of Milano querying an AllegroGraph service with a geo-spatial query to find the path in this graph (test presented in Section 4.2);
 - o workflow 2b: presented in D6.3, Section 2.5.4, it uses a hybrid approach: it loads the graphs around the start and the goal node with AllegroGraph and connects them by loading the graph of the main roads of Milano contained in an RDF file (test description in Section 4.3);
 - o rule-based path finding workflow: based on OntoBroker, it processes the path using a rule-engine system (this test will be presented Section 4.4).
- *Stress tests on Urban Computing workflows*: these tests are complementary to the previous ones, in fact we verified how LarKC performance is influenced by the number of concurrent requests that it receives at the same time. As explained in D6.5, for the Urban LarKC we developed some workflows, and as we will see in the following, these workflows have different peculiarities; it means that they stress the LarKC platform in different ways, so we considered every kind of workflow separately from the others and we performed independent stress tests. Summarizing, the workflows we considered are:
 - o one to process the most desirable path between two nodes in a graph using the Operational Research approach (Section 4.5);
 - o one to find the monuments in Milano accessing LOD and DBPedia (Section 4.6);
 - o one to get the events located in Milano from Eventful, a Web 2.0 Web site where users publish and manage events (Section 4.7).
 - o one to process the most desirable path using the rule-based approach (Section 4.8).
- *Urban Computing data quality evaluation*: these tests are done in order to obtain indicators related to the Urban Computing scenario. We want to obtain two main goals: the first is obtaining qualitative indications about the state of the application and the second is to build an initial dataset to make comparisons with performance data that we will collect in next tests (and we will present in next data and performance deliverables). We considered the following two workflows:
 - o Monument workflow: retrieve Milano monuments collecting data from the LOD (Section 4.9);
 - o Event workflow: get the Milano events from a Web 2.0 event site (Section 4.10).

In the next sections we will describe each test we performed. The general structure of every test report is the following:

- test goals: explain the reasons we considered to perform the test.;
- considered workflow: a description of the workflow involved in the test;
- methodology: an explanation of how we conducted the test;
- environment: the hardware and the software used;
- results: the results we obtained;
- considerations: a brief analysis of the results obtained in the test.

Finally, in Section 4.11 we give an outlook on the future evaluation methods for the traffic prediction.

4.1. Operational Research path finding performance test – Workflow 1a

4.1.1. Test goals

In this test we collected data about the execution of LarKC with the path finding workflow 1a (described in D6.3) monitoring the execution of the application in order to catch some relevant time values. This information allows us to understand the difference between the complete system execution time and plug-in execution time measuring the non-plug-in execution time, being *overhead*:

- the initial query processing;
- the construction of the query output;
- the time required for the exchange of data between a plug-in and its successor;
- the building of the workflow;
- etc.

4.1.2. Considered workflow

The path finding workflows were described in D6.3. In particular, the workflow used in this test is the one represented in Figure 1: an identifier locates the position of an RDF file stored on a remote server and loads it, a Selector prepares the loaded RDF model in a suitable format for the last plug-in of the workflow, a Reasoner, that executes a SPARQL query and returns the results.

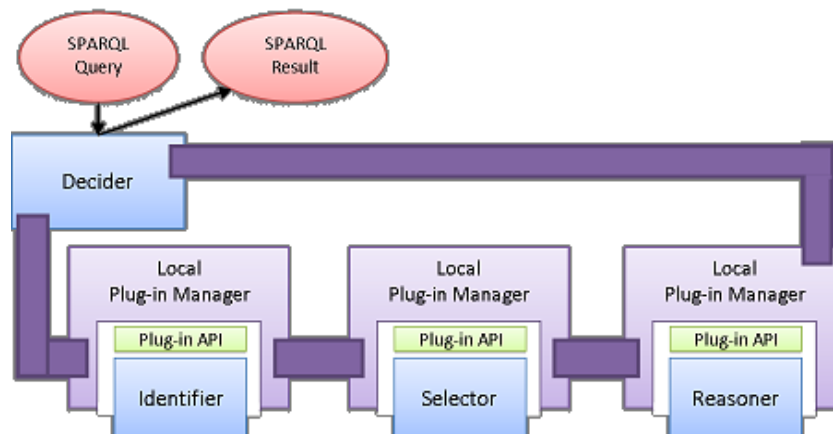


Figure 1 - Path finding workflow 1a

When this workflow is executed, an RDF graph stored in a XML file and located on a remote server is downloaded completely. In this test we used the RDF graph of the historical centre of Milano, a subset of the whole Milano graph.

4.1.3. Methodology

In order to execute the performance test on the workflow 1a (and more generally on the Operational Research path finding workflows) we used a profiling platform, Test & Performance Tools Platform³ (TPTP), an Eclipse plug-in. It is configurable with the kind of information that should be collected (for example time, memory, class interactions, etc.) and the name of classes and methods that should be monitored. This tool slows the execution of the application, but by analyzing only a subset of methods (as we did) the delay is not perceptible.

When we started the design and development of the Alpha version of Urban LarKC we implemented a demo application using “standard” Semantic Web tools simulating the behaviour that the LarKC workflow was intended to show. We used this application as a term of comparison (we will call it Operational-research Term of Comparison or *OR-ToC*, to differentiate it from another term of comparison used in other tests): we monitored this application with the same TPTP tool getting information about its execution time and we compared it with the alpha Urban LarKC, mainly to understand how much overhead is introduced by LarKC.

An important thing that we had to consider was that both the Urban LarKC and the OR-ToC should be executed in the same environment. To ensure this, both applications:

³ <http://www.eclipse.org/tptp/>



- were executed on the same machine (see below);
- collected the data interacting with the same data sources;
- were restarted before each new execution.

We decided to consider a set of queries (instead of one only) in order to collect data allowing us to understand how the workflow performance is related to the input query. The chosen queries are very similar, the difference is given by the start and end nodes that vary between queries, as explained in Table 1.

| Query number | Start node | End node | Note |
|--------------|------------|----------|--|
| 1 | 8742 | 8712 | The nodes are very near (there is only one link between them) |
| 2 | 19075 | 8120 | The nodes are moderately far apart (they are both in the centre of Milano) |
| 3 | 31122 | 11812 | The nodes are far apart (they are both outside the Milano centre) |
| 4 | 5940 | 8136 | The first node doesn't exist (so the path could not be found) |

Table 1 - Pairs of nodes used in the performance tests

In every execution we considered the whole path finding workflow execution, including the download and the processing of the XML/RDF file with the Milano topology model stored remotely.

In the following we describe the schema we used to execute the test. In order to obtain accurate results, we executed the test following this schema:

- for each query:
 - o we send three times the query to the Urban LarKC collecting the relative time values;
 - o we send the query to OR-ToC three times (collecting the times);
 - o we calculated the mean of the collected time values obtaining several average time values (the overall average time, the average time required by the identifier, etc.) for both the Urban LarKC and the OR-ToC.

4.1.4. Environment

Hardware

Processor: Intel Xeon 3.60/3.60 GHz (two dual core CPUs)

RAM: 4 GB

Operating system: Microsoft Windows Server 2003 Standard Edition 32 bit (Service Pack 2)

Software

LarKC:

Revision: 744 (August 25, 2009 2:35 PM)

JVM Heap Memory: 512 MB

4.1.5. Test results

We submitted to this workflow three queries: Query 1, Query 2 and Query 4 (see Table 1): Query 4 contains a node that doesn't exist, so it's not contained in the RDF graph that is loaded. Query 3 contains two nodes that are not contained in the graph of the historical centre of Milano, so the behaviour in this case is the same of Query 4, and we choose not to use this query.

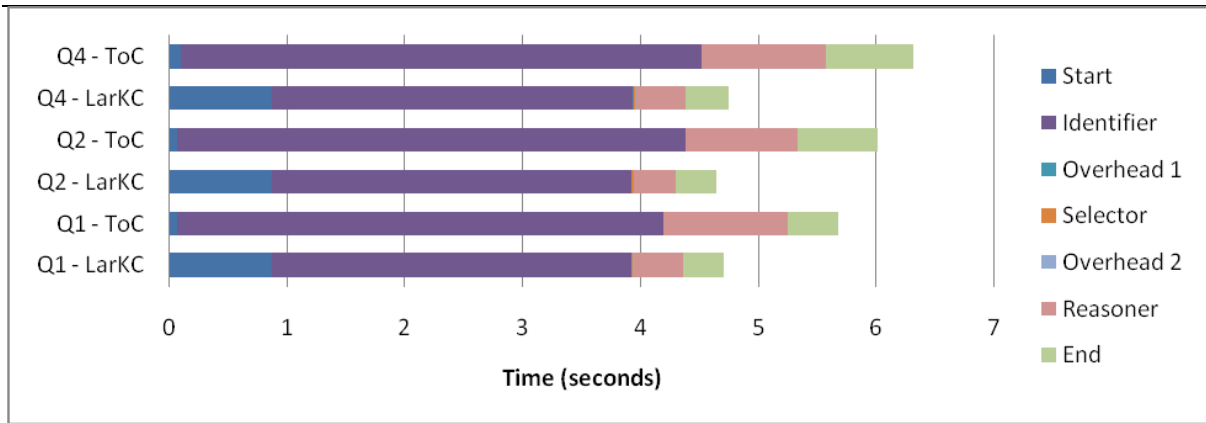


Figure 2 – Graph of execution times in workflow 1a

Figure 2 shows the results of experiments on workflow 1a: we identified the following intervals:

- Start: the time between the reception of the query and the start of the Identifier task;
- Identifier: the time required to execute the task of this plug-in (it's the time used by identify method)
- Overhead 1: the time between the end of the Identifier and the start of the Selector;
- Selector: the time used by the Selector plug-in to complete its task (the execution time of select method);
- Overhead 2: the time between the end of the Selector and the start of the Reasoner;
- Reasoner: the time required to complete sparqlSelect method: the conversion of the RDF graph into a JGraphT graph, the execution of Dijkstra algorithm and the construction of the VariableBinding that should be returned;
- End: the time used by the application to compose the result, to send it back to the requester and to terminate the workflow execution.

| | Query 1 | | Query 2 | | Query 4 | |
|------------|------------|-----------|------------|-----------|------------|-----------|
| | Q1 - LarKC | Q1 - ToC | Q2 - LarKC | Q2 - ToC | Q4 - LarKC | Q4 - ToC |
| | seconds | seconds | seconds | seconds | seconds | seconds |
| Start | 0.8668196 | 0.0608633 | 0.8649681 | 0.0630733 | 0.8713301 | 0.0971243 |
| Identifier | 3.0469137 | 4.1302727 | 3.0500078 | 4.318763 | 3.0614435 | 4.415803 |
| Overhead 1 | 0.0043728 | 0.0006423 | 0.0042398 | 0.0005607 | 0.0041373 | 0.0005243 |
| Selector | 0.0149915 | 0 | 0.0151222 | 0 | 0.0151621 | 0 |
| Overhead 2 | 0.0041965 | 0 | 0.0042995 | 0 | 0.0040259 | 0 |
| Reasoner | 0.4196175 | 1.0490287 | 0.3529319 | 0.9431153 | 0.4169753 | 1.051417 |
| End | 0.3435999 | 0.4288253 | 0.3434285 | 0.6778293 | 0.3723335 | 0.750199 |
| Overhead | 1.2189887 | 0.490331 | 1.2169359 | 0.7414633 | 1.2518269 | 0.8478477 |
| Total | 4.7005113 | 5.6696323 | 4.6349977 | 6.0033417 | 4.7454077 | 6.3150677 |

Table 2 - Execution times in workflow 1a

In Table 2 it's possible to find the average execution times of the different components that we collected in our test.

4.1.6. Considerations

Surprisingly, in this case LarKC is faster than the OR-ToC: it's possible to notice that the main difference is given by the identifier task that in the OR-ToC requires a bigger amount of time. The cause of this could be that the OR-ToC has been realized using Jena, while LarKC uses other technologies (Sesame API, LarKC Data Layer and so on). In Figure 2 Selector and Overheads 1 and 2 timings are not visible, because the time required by them is very small, as it's possible to see in Table 2 that contains all the means of the times we collected.

In the OR-ToC the Selector part is missing, so the Selector and Overhead 2 times are set to 0. In the row Overhead is reported the sum of the overheads: Start, Overhead 1, Overhead 2 and End. We

represented this value graphically in Figure 3: it's possible to observe that the overhead introduced by LarKC is greater than the one of OR-ToC, but the difference is very little (less than one second).

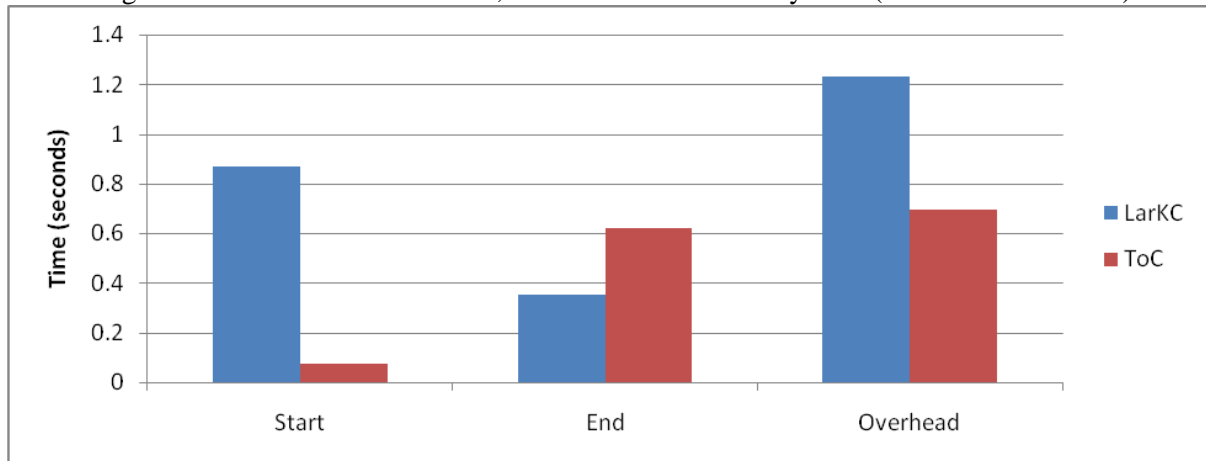


Figure 3 - Overheads of LarKC and OR-ToC in workflow 1a

Lastly, from the collected values we can observe that the time required to find a response to the input query doesn't vary significantly with the different queries.

4.2. Operational Research path finding performance test – Workflow2a

4.2.1. Test goals

The objectives of this test are the same of the one presented above; see Section 4.1.1 for more details.

4.2.2. Considered workflow

Figure 4 shows the workflow 2a, described in D6.3: a Transformer processes the input query extracting geographical information from it, an Identifier uses this information in order to interact with an AllegroGraph service to get a graph containing both the start and the goal nodes, then the Selector and the Reasoner process the collected data in order to find a path.

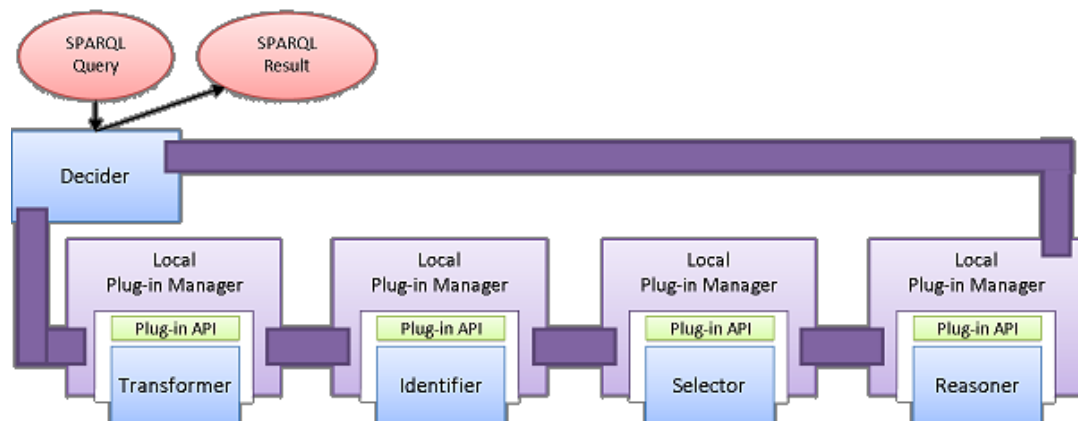


Figure 4 - Path finding workflow 2a

4.2.3. Methodology

The methodology is the same of the one described in the previous test. The description is presented in Section 4.1.3.



4.2.4. Environment

Hardware

Processor: Intel Xeon 3.60/3.60 GHz (two dual core CPUs)

RAM: 4 GB

Operating system: Microsoft Windows Server 2003 Standard Edition 32 bit (Service Pack 2)

Software

LarKC:

Revision: 744 (August 25, 2009 2:35 PM) JVM Heap Memory: 512 MB

4.2.5. Test results

We tried all the four queries described above but we noticed that Query 4 (the one with a node that doesn't exist) causes an exception in the Identifier plug-in that it's not managed properly, this is why we didn't consider Query 4 in this test.

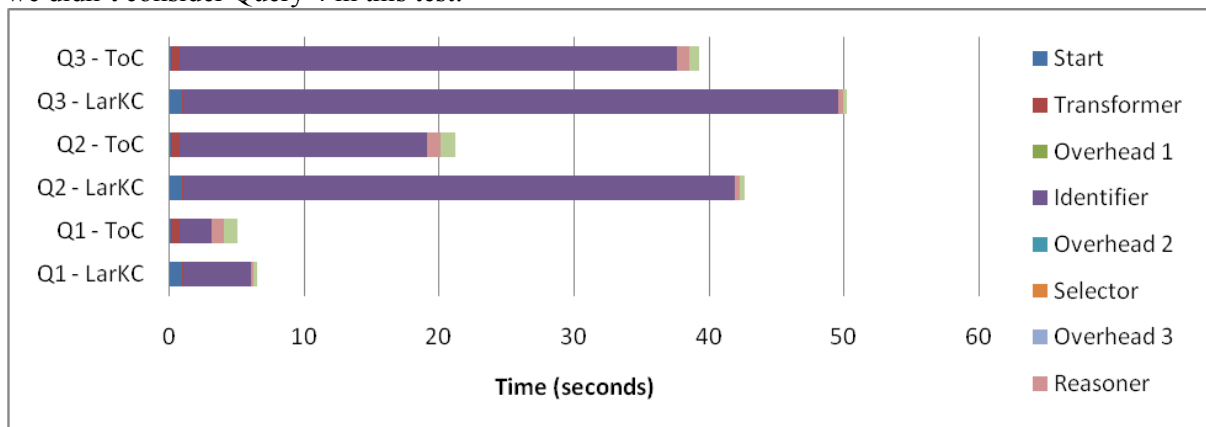


Figure 5 - Graph of execution times in workflow 2a

Figure 5 shows the results we obtained in our test on workflow 2a. The intervals we identified collecting data are similar to the ones we described in Section 4.1.5; the main differences are given by the introduction of Transformer plug-in, that introduces two new intervals:

- Transformer: the time required to process the input query (it's the time used by `transform` method)
- Overhead 1: the time between the end of the Transformer and the start of the Identifier.

In Table 3 the average time values are shown. A thing that we have to report is that the processing of Query 3 completes regularly but no valid paths are found: we didn't find the cause of this behaviour of the application and we will try to investigate and to fix this problem.

| | Query 1 | | Query 2 | | Query 3 | |
|-------------|------------|-----------|------------|-----------|------------|-----------|
| | Q1 - LarKC | Q1 - ToC | Q2 - LarKC | Q2 - ToC | Q3 - LarKC | Q3 - ToC |
| | seconds | seconds | seconds | seconds | seconds | seconds |
| Start | 0.878646 | 0.0641598 | 0.8747324 | 0.0635357 | 0.8759973 | 0.0619414 |
| Transformer | 0.133121 | 0.6599378 | 0.1343683 | 0.6716549 | 0.1363907 | 0.6578656 |
| Overhead 1 | 0.0041158 | 0.0040085 | 0.0041414 | 0.0039045 | 0.0040871 | 0.0039239 |
| Identifier | 4.9741037 | 2.3704303 | 40.860423 | 18.35078 | 48.57999 | 36.871591 |
| Overhead 2 | 0.0039231 | 0.0007141 | 0.0037014 | 0.000693 | 0.0036844 | 0.0008137 |
| Selector | 0.0101131 | 0 | 0.0099005 | 0 | 0.0102752 | 0 |
| Overhead 3 | 0.0038948 | 0 | 0.0039481 | 0 | 0.0038491 | 0 |
| Reasoner | 0.1640335 | 0.9112142 | 0.3803811 | 1.0431571 | 0.3087569 | 0.9674903 |
| End | 0.3384583 | 0.9934464 | 0.3638801 | 1.0317477 | 0.32905 | 0.6849145 |
| Overhead | 1.2290379 | 1.0623287 | 1.2504034 | 1.0998809 | 1.2166678 | 0.7515936 |
| Total | 6.5104093 | 5.003911 | 42.635477 | 21.165473 | 50.25208 | 39.248541 |

Table 3 - Execution times in workflow 2a (with Query 3 no valid paths are obtained)

4.2.6. Considerations

In this case the situation is opposite to the previous one. In fact LarKC is slower than OR-ToC, as it's also possible to see in Table 3: the difference between OR-ToC and LarKC varies a lot (in Query 2 the difference is more than 20 seconds).

In a similar way to results we obtained in workflow 1a (Section 4.1.5), in this case the part that requires more time is the Identifier. In addition, the time required to process the query varies when the query changes: the further the two points are apart, the more the time the Identifier requires to complete its task. The cause is the AllegroGraph service, which finds an answer quickly only when two nodes are close together.

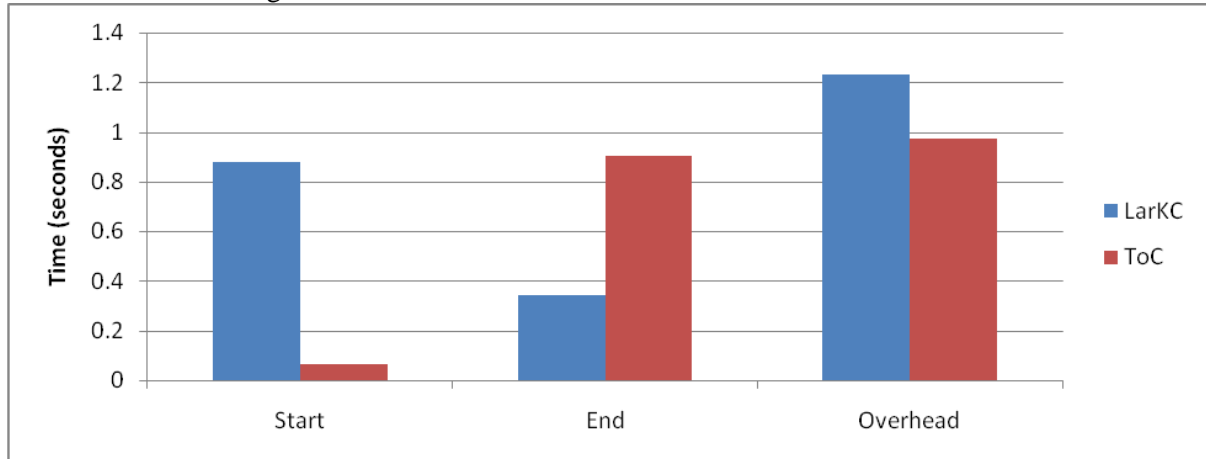


Figure 6 - Overheads of LarKC and OR-ToC in workflow 2a

Figure 6 shows the overhead of LarKC and the OR-ToC; the results we obtained are similar to the workflow 1a results: LarKC is a bit slower than OR-ToC (they are very close).

4.3. Operational Research path finding performance test – Workflow 2b

4.3.1. Test goals

The goals of this test are the same of the two tests we presented below. See Section 4.1.1 to get more information.

4.3.2. Considered workflow

Workflow 2b (Figure 7) is similar to workflow 2a, in fact there is a Transformer plug-in as first component of the workflow, that analyzes the input query and extract geographical information from it (it's required by the following Identifier). The main difference is the Identifier: in fact, in this case the identification policy is an intermediate case between the first two workflows we analyzed: it collects data interacting with both an AllegroGraph service (it selects graphs around start and goal nodes) and a remote server containing an RDF file (the main roads).

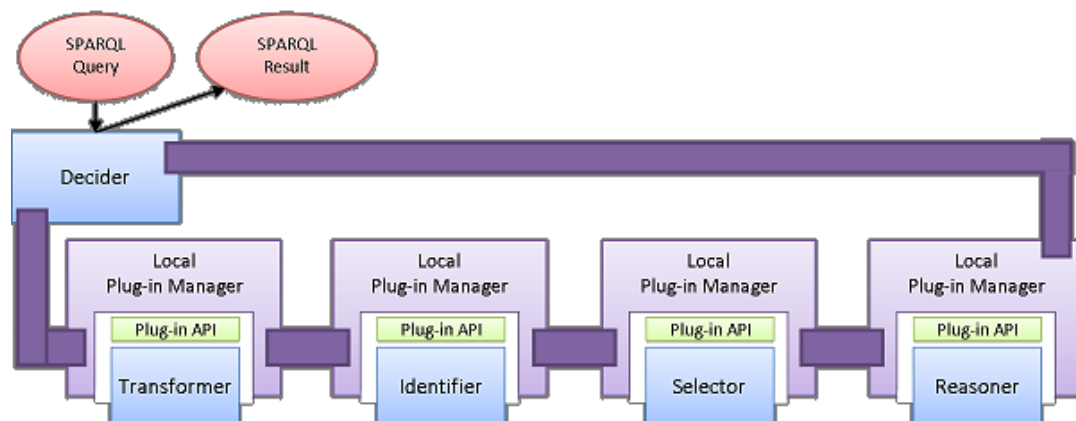


Figure 7 - Path finding workflow 2b



4.3.3. Methodology

A description of the methodology we followed in this test is available in Section 4.1.3.

4.3.4. Environment

Hardware

Processor: Intel Xeon 3.60/3.60 GHz (two dual core CPUs)

RAM: 4 GB

Operating system: Microsoft Windows Server 2003 Standard Edition 32 bit (Service Pack 2)

Software

LarkC:

Revision: 744 (August 25, 2009 2:35 PM)

JVM Heap Memory: 512 MB

4.3.5. Test results

Similar to workflow 2a, Query 4 raises an exception that is not caught and managed properly by the Identifier plug-in (the code is the same as the plug-in used in workflow 2a), so we didn't consider it.

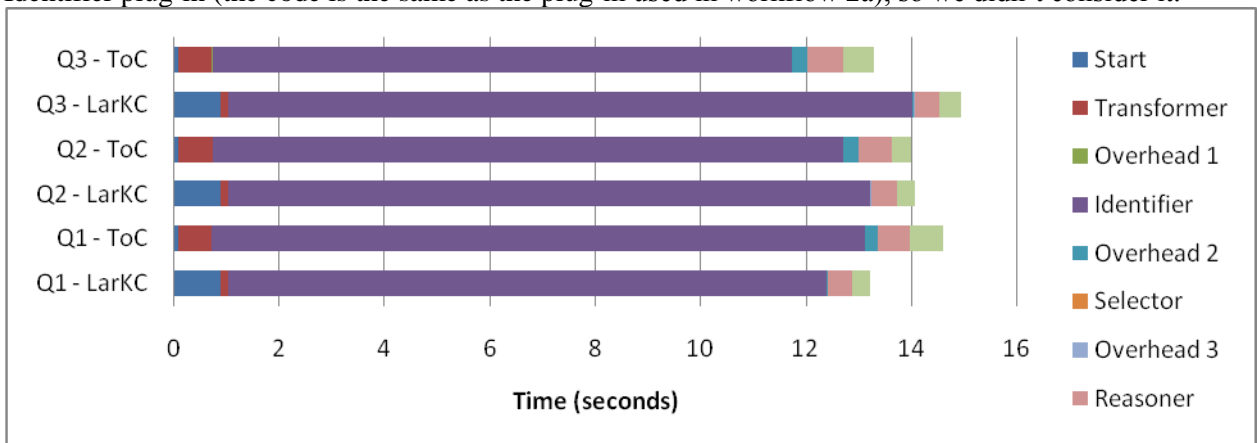


Figure 8 shows the average execution times we collected: the chart seems to confirm that it is the intermediate case: the results of LarKC and OR-ToC are very similar and neither is consistently better than the other.

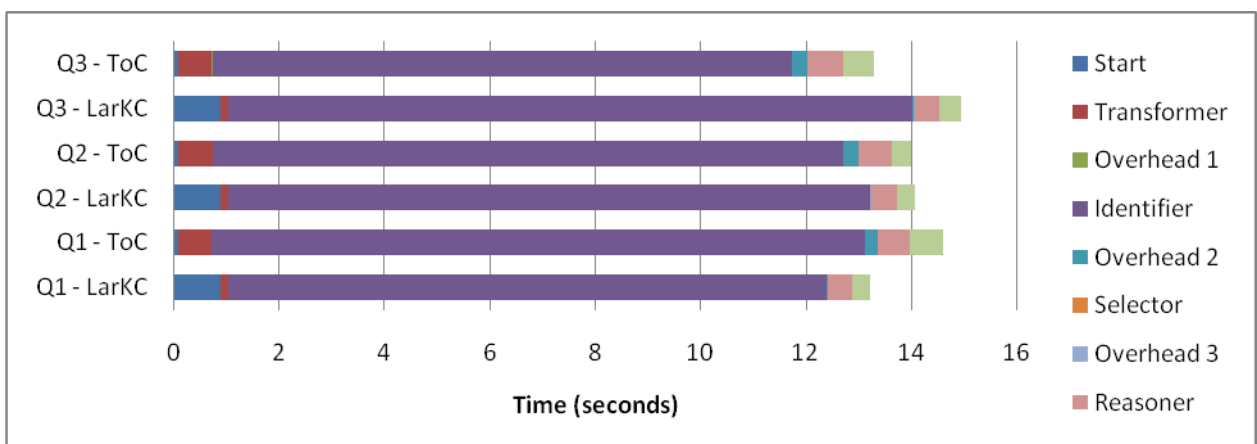


Figure 8 - Graph of execution times in workflow 2b



In Table 4 it's possible to find the values that every part of the workflow requires to complete its task. The processing of Query 2 didn't return a valid path: it happens because the selected graphs around start and goal nodes are too small and they didn't connect to main roads.

| | Query 1 | | Query 2 | | Query 3 | |
|-------------|------------|-----------|------------|-----------|------------|-----------|
| | Q1 - LarKC | Q1 - ToC | Q2 - LarKC | Q2 - ToC | Q3 - LarKC | Q3 - ToC |
| | seconds | seconds | seconds | seconds | seconds | seconds |
| Start | 0.8791333 | 0.0633325 | 0.8791581 | 0.0623544 | 0.8839545 | 0.0633027 |
| Transformer | 0.136858 | 0.6510789 | 0.1384766 | 0.6560691 | 0.1364619 | 0.6517515 |
| Overhead 1 | 0.0040717 | 0.0039995 | 0.0039567 | 0.0039651 | 0.0041346 | 0.0038972 |
| Identifier | 11.376961 | 12.396858 | 12.194041 | 11.980688 | 13.01038 | 11.020246 |
| Overhead 2 | 0.0186046 | 0.2425914 | 0.0185196 | 0.3001017 | 0.0168702 | 0.2981816 |
| Selector | 0.0003298 | 0 | 0.0003318 | 0 | 0.0003305 | 0 |
| Overhead 3 | 0.0140246 | 0 | 0.0139317 | 0 | 0.0121145 | 0 |
| Reasoner | 0.4575351 | 0.6282559 | 0.4788448 | 0.6305161 | 0.4848937 | 0.6737885 |
| End | 0.3429683 | 0.6176933 | 0.3455543 | 0.3805608 | 0.4058699 | 0.5835856 |
| Overhead | 1.2588026 | 0.9276167 | 1.2611204 | 0.746982 | 1.3229436 | 0.9489671 |
| Total | 13.230487 | 14.603809 | 14.072815 | 14.014255 | 14.955009 | 13.294753 |

Table 4 - Execution times in workflow 2b (with Query 2 no valid paths are obtained)

4.3.6. Considerations

From the results we obtained we can observe that most of the time required by every workflow execution is used by the Identifier (more than 66% of the overall time). This is a behaviour similar to the one obtained in previous tests.

Additionally, the overhead introduced by LarKC before and after the workflow execution is not significant compared with the one obtained by OR-ToC, as it's possible to see in Figure 9 (the values are very similar with the ones described below).

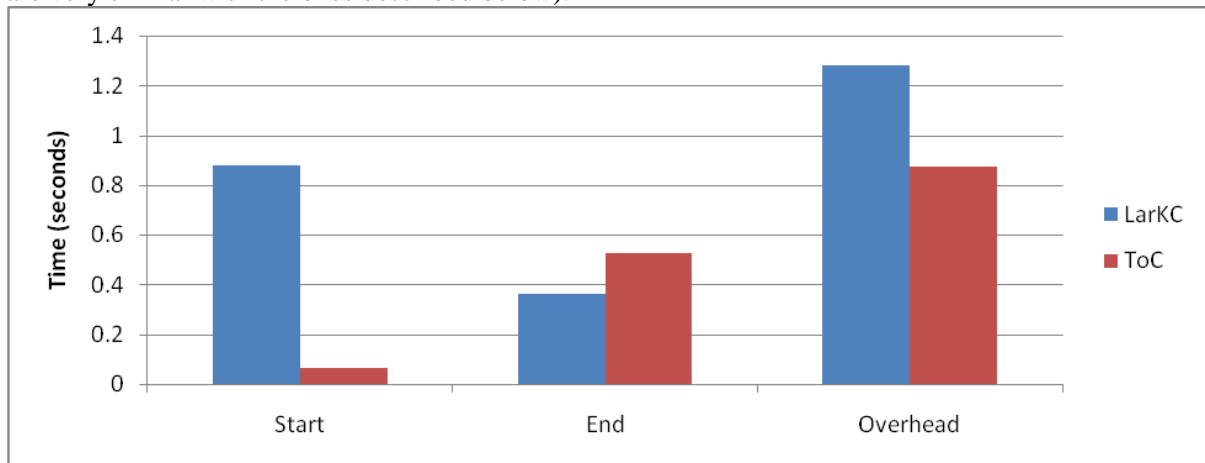


Figure 9 - Overheads of LarKC and OR-ToC in workflow 2b

4.4. Rule-based path finding performance test

4.4.1. Test goals

In order to evaluate Urban LarKC with an existing commercial product, we selected Ontobroker from Ontoprise, because it has shown good performance with the U-City project which has been conducted by Saltlux in the past. The U-city project is similar to the Urban Computing use case of LarKC in that it tries to find the most desirable path with traffic information under various road conditions. U-city was implemented using Ontobroker to reason with F-logic rules and F-logic queries to find paths.

In LarKC, we developed an Ontobroker wrapper, with the help of STI Innsbruck, to use Ontobroker as a Reasoner plug-in for the LarKC platform, thus taking advantage of an existing commercial

product - Ontobroker converts an input RDF graph into F-Logic facts and rules. This LarKC plugin uses the Bellman-Ford algorithm to find the shortest path.

The goal of this test is to compare the performance of a LarKC workflow that makes use of the aforementioned plug-in with an application that makes direct use of OntoBroker (from now on indicated as Rule-based Term of Comparison or RB-ToC). This comparison can help us understand the impact and overhead of the LarKC platform when using it in an Urban Computing application as well as the feasibility of integrating a commercial reasoner within LarKC.

4.4.1. Considered workflow

The workflow used in this test is graphically represented in Figure 10 together with the rule-based Term of Comparison. The workflow consists in an Identifier plug-in, which takes the Milano roads RDF, followed by a Selector plug-in, which passes on the selected data, and finally a Reasoner plug-in (OBLarKCShortestPathReasoner) where the actual computation for the shortest path takes place. This Reasoner plug-in wraps an OntoBroker instance, in which the RDF input data is translated to F-logic and loaded in the knowledge base; the rules to apply the Bellman-Ford algorithm were previously coded and added to the OntoBroker instance.

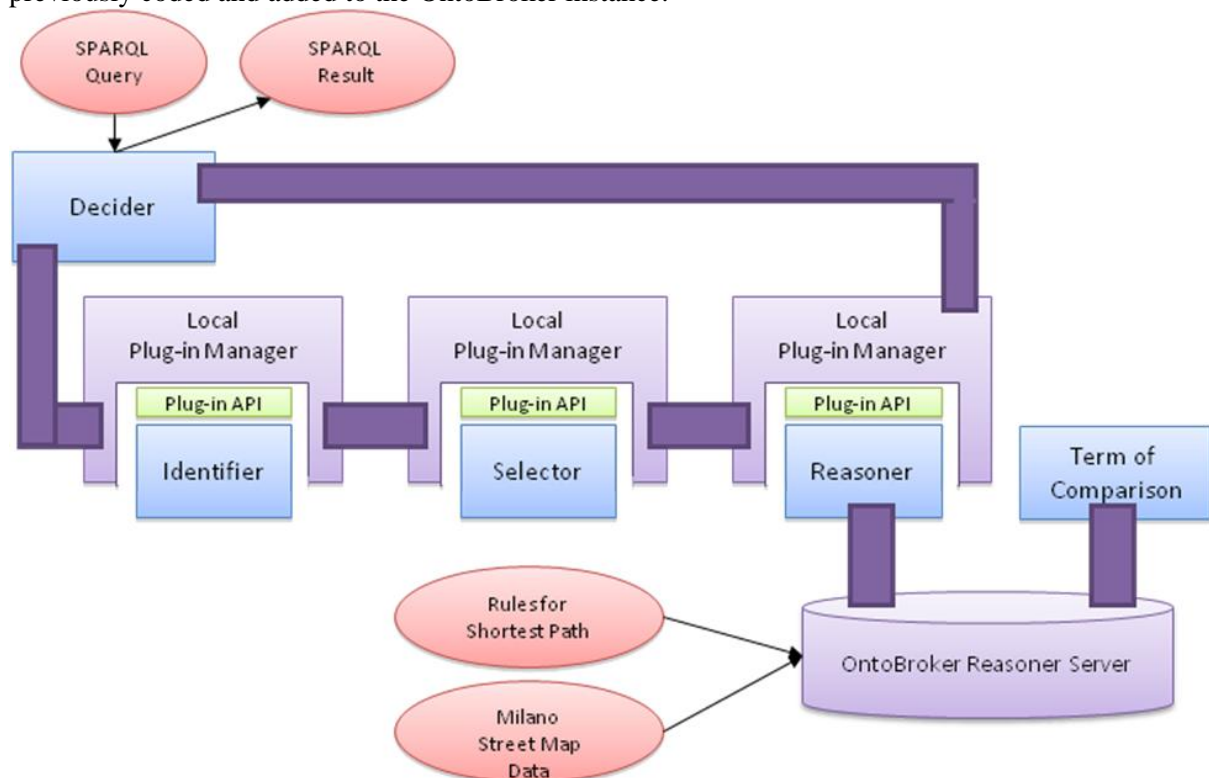


Figure 10 – Workflow used to test the rule-based path-finding approach.

The Term of Comparison application, on the other hand, is a custom application that directly accesses the OntoBroker instance with the same data and the same rules.

4.4.2. Methodology

In a similar way to the comparison tests described in Sections 4.1-4.2-4.3 and 4.4, we focused our attention on two kinds of tests: one, described hereafter, is a “comparison test by data increase”, while the second – described in Section 4.8 – is a “comparison test by user increase”.

The goal of the current test is to verify how LarKC performance measurements are influenced by the number of links and nodes required to find the path and comparing them to the RB-ToC ones.

We made some tests with increasing the numbers of links and nodes composing the most desirable path. As nodes for points of network are increased, OntoBroker needs more steps to generate relevant queries. We used 10 kinds of queries which varied in minimum link count from 1 to 10 for the shortest path. The OntoBroker reasoner generates OutOfMemory errors when the number of nodes required are over 11 due to the data size. Only limited amount of data loading is possible with the memory loading option for OntoBroker. This limitation could be overcome in the future by using persistent storage option with loss of speed. We are going to test this option in the next deliverable.



4.4.3. Environment

Hardware

Processor: Intel Pentium 4 3.06 GHz
 RAM: 3.25 GB
 HDD: 41 GB
 Operating system: Microsoft Window Server 2003 R2 32bit (Service Pack 2)

Software

Alpha Urban LarKC
 Revision: 744 (August 25, 2009 2:35 PM)
 JVM Heap Memory: 512 MB

OntoBroker
 Version: Enterprise 5.2 Update 1
 JVM Heap Memory: 1.5 GB

Ontology

Milan Street Map Ontology
 Name: ama-xml.flo
 Format: F-logic
 Size: 127MB
 Triple Count : 1,184,000 (not inferred)
 Remarks: ama-xml.flo is converted by OntoBroker tools from ama-xml.rdf

A variant Bellman-Ford algorithm Rules
 Name: shortestPathFlogicRule.flo
 Format: F-logic
 Size: 2KB
 Rule Count: 7

4.4.4. Test results

The results of the execution of the path finding calculation by increasing the number of links between the starting and ending nodes are represented in the following table.

| Minimum link count for the shortest path | Total workflow execution (sec) | LarKC overhead (sec) | Identifier execution (sec) | OntoBroker wrapper plug-in execution (sec) | Rule-based ToC execution (sec) |
|--|--------------------------------|----------------------|----------------------------|--|--------------------------------|
| 1 | 100,78 | 0,04 | 100,30 | 0,44 | 0,80 |
| 2 | 106,94 | 0,06 | 105,41 | 1,47 | 1,92 |
| 3 | 115,06 | 0,07 | 111,27 | 3,72 | 4,00 |
| 4 | 116,44 | 0,05 | 110,09 | 6,30 | 6,75 |
| 5 | 100,34 | 0,08 | 88,84 | 11,42 | 10,30 |
| 6 | 160,14 | 0,03 | 145,61 | 14,50 | 16,67 |
| 7 | 178,98 | 0,04 | 155,22 | 23,72 | 24,17 |
| 8 | 159,09 | 0,06 | 123,50 | 35,53 | 35,41 |
| 9 | 181,72 | 0,08 | 125,50 | 56,14 | 55,94 |
| 10 | 216,89 | 0,05 | 129,31 | 87,53 | 87,03 |

Table 5 - Results of performance with data increase for finding the shortest path

From the values presented in Table 5, we can easily understand that loading the data into OntoBroker at run time (via the use of the Identifier plug-in) is the major bottleneck in the workflow execution, while the overhead due to the platform-specific execution (starting and ending the pipeline, passing the data between the plug-ins, etc.) is present but its impact is little. The Rule-based ToC, in which the



data and rules were loaded in advance, can therefore be compared with the Reasoner plug-in: the two components show a similar behaviour.

4.4.5. Considerations

The current version of the LarKC platform allows for on-the-fly processing of queries and data. This scenario, however, does not apply to cases in which a large part of the processing can be made at “batch-time”, before the actual processing of user queries. In the specific case described above, the slowness of the workflow execution is due to a step of loading the data and materializing the inference within the OntoBroker reasoner. Therefore, our suggestion would be to include a different kind of plug-ins to deal with this “batch-time” processing. Ideally, to wrap OntoBroker, new Identifier, Selector and Transformer plug-ins, specifically designed in advance to work with OntoBroker, should be needed.

4.5. Alpha Urban LarKC stress test – Path finding workflow

4.5.1. Test goals

The main objective of this test is to understand how the Alpha Urban LarKC behaviour is influenced by the number of concurrent requests it have to process. It’s important to test and to collect this kind of information in order to obtain objective indicators about the state of the Alpha Urban LarKC and to make useful considerations about the application and more generally about LarKC.

In this first stress test we considered a “limited” environment (all the machines are located in a LAN). This allows us to have a high control on all the machines involved in the test. This can be considered as a constraint, in fact LarKC is designed and developed in order to work in the Web; in following tests (Section 4.6 and 4.7) the constraint is relaxed and LarKC interacts with data sources located on the Web.

4.5.2. Considered workflow

In D6.3 we described different workflows to evaluate the most desirable path (we used them in the LarKC performance tests described above). In the actual version of the Alpha Urban LarKC we used only the workflow 1a (Figure 11), loading the whole map of Milano. We modified the Identifier plug-in in order to load the RDF model of the whole Milano map only at the first request that LarKC receives: when following requests arrive the same RDF graph is used and it is sent to the Selector-Reasoner plug-ins (it’s something similar to a cache). In this way the workflow can be executed faster (some seconds instead of about one minute). In this test we will consider the workflow executions following the first, when the RDF model has been loaded already.

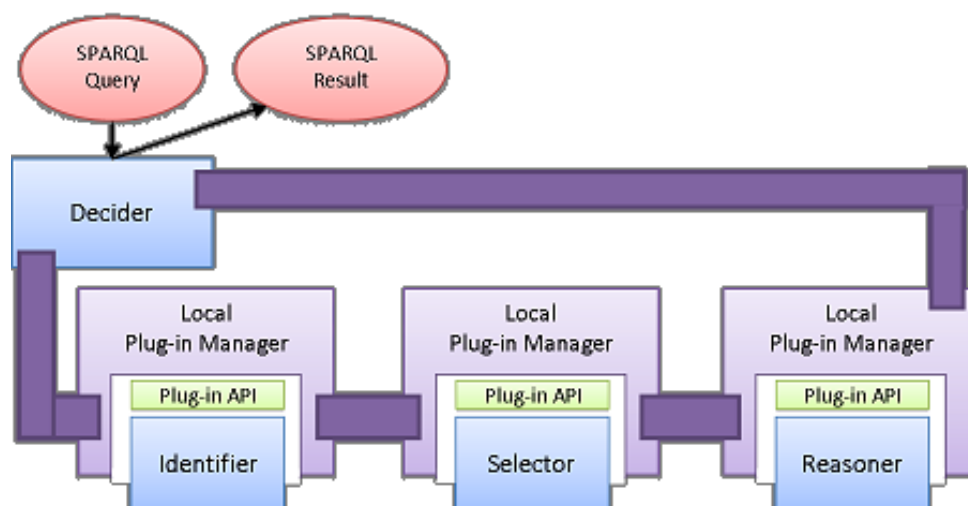


Figure 11 - Path finding workflow

We analyzed the execution of the path finding workflow and we measured how much time each plug-in uses to complete its job. Table 6 shows the percentage of total time that every plug-in requires to execute its job.

| Plug-in | Execution percentage | Execution time |
|------------|----------------------|----------------|
| | | seconds |
| Identifier | 1.57% | 0.04349 |
| Selector | 0.01% | 0.00027 |
| Reasoner | 98.42% | 2.72263 |

Table 6 - Execution time allocation for path finding workflow plug-ins

It's clear that the main operation in this case is the Reasoner task: it converts the RDF model into a suitable form for processing and it applies Dijkstra's algorithm to it. It means that in this case the workflow execution mainly stresses the processors. This is an important fact that we should consider when we will analyze the results of the test.

4.5.3. Methodology

In order to submit requests to LarKC we wrote a program: it requires an initial set of requests that can be provided via XML file, allowing an easy scheduling of the tests; alternatively it's possible to select one of several default requests generators that we built. A *request* is a triple composed by an identification code, a SPARQL query and a submission time (in order to allow delayed requests). The application can send *multiple requests* (set of one or more requests sent at the same time) to the Urban LarKC recording the response time for each of them. The act of sending a multiple request (in other words the submission a set of queries to LarKC SPARQL end-point at the same time) is an *experiment*.

This test has been structured as follows:

- for each multiple request (characterized by a different number of requests):
 - o we executed three experiments and for each of them we calculated the average times;
 - o we calculated the mean and the variance of the average time values of each experiment (this mean is the *average response time* of the multiple request).

4.5.4. Environment

For the execution of this test we used the environment showed in Figure 12. It is a system composed by three machines connected among them in a cabled local network.

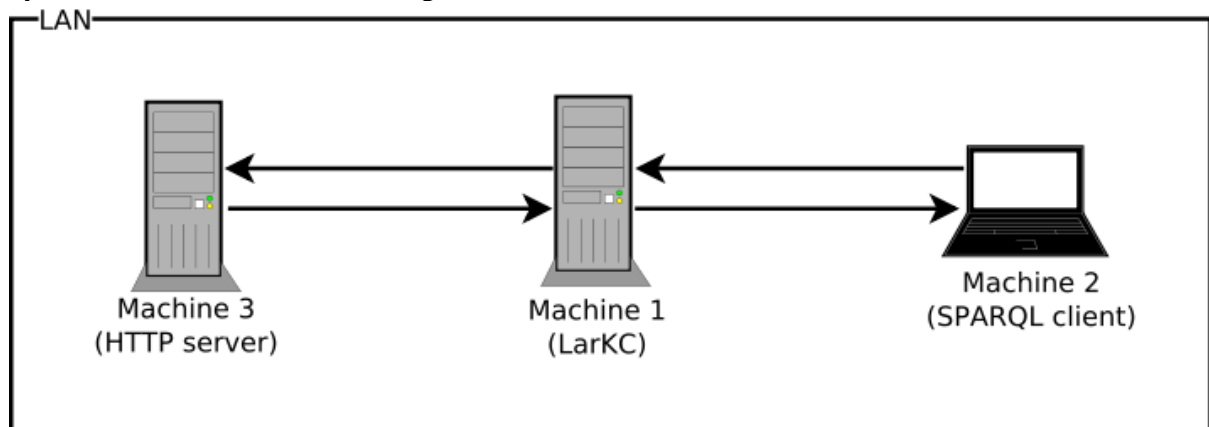


Figure 12 - Enviroment used in the path finding stress test

Machine 1 is the one where LarKC was executed; its configuration is the following:

- Processor: Intel Xeon 3.60/3.60 GHz (two dual core CPUs)
- RAM: 4 GB
- Operating system: Microsoft Windows Server 2003 Standard Edition 32 bit (Service Pack 2)
- LarKC:
 - o Revision: 744 (August 25, 2009 2:35 PM)
 - o JVM Heap Memory: 512 MB

We used Machine 2 to send requests to LarKC (with the application described below). It's configuration is the following:

- Processor: Intel Core 2 2.16 GHz
- RAM: 2 GB



- Operating system: Microsoft Windows XP Professional (Service Pack 3)
 Machine 3 was used as HTTP server to supply the Milano graph RDF file, its configuration is the same of Machine 1.

4.5.5. Test results

In this test we started considering all the multiple requests composed by a number of requests between 1 and 25. Considering the values we obtained, we choose to execute some other tests with multiple requests composed by 30, 40 and 50 requests. In Table 7 it's possible to observe the average response times of the multiple requests and relative variances we calculated starting from the data collected in the tests.

| Number of requests | Mean | Variance | Number of requests | Mean | Variance |
|--------------------|----------|----------------------|--------------------|----------|----------------------|
| | seconds | seconds ² | | seconds | seconds ² |
| 1 | 2.277688 | 0.004915 | 15 | 23.99613 | 0.021756 |
| 2 | 2.68147 | 0.001414 | 16 | 26.24856 | 0.157506 |
| 3 | 4.584027 | 0.00019 | 17 | 27.48328 | 0.130953 |
| 4 | 6.439797 | 0.001745 | 18 | 30.40831 | 0.32665 |
| 5 | 7.952923 | 0.000221 | 19 | 30.90711 | 0.159221 |
| 6 | 10.12021 | 0.267691 | 20 | 33.46662 | 0.410749 |
| 7 | 11.52904 | 0.070279 | 21 | 34.3479 | 0.285833 |
| 8 | 12.75063 | 0.005113 | 22 | 36.23883 | 1.726706 |
| 9 | 14.48108 | 0.028805 | 23 | 37.52821 | 0.004532 |
| 10 | 16.28992 | 0.199104 | 24 | 39.18142 | 0.150017 |
| 11 | 18.22337 | 0.221386 | 25 | 41.01503 | 0.8688 |
| 12 | 19.36423 | 0.240818 | 30 | 49.97639 | 0.561443 |
| 13 | 21.26006 | 0.166341 | 40 | 67.07202 | 0.486032 |
| 14 | 23.50564 | 0.095102 | 50 | 87.26458 | 1.478979 |

Table 7 - Results of multiple requests with path finding workflow

In Figure 13 and in Figure 14 a graphical visualization of the average response times is available: the first chart shows the trend of the multiple requests with a number of requests between 1 and 25, the second one shows the trend in a larger range (1-50 requests).

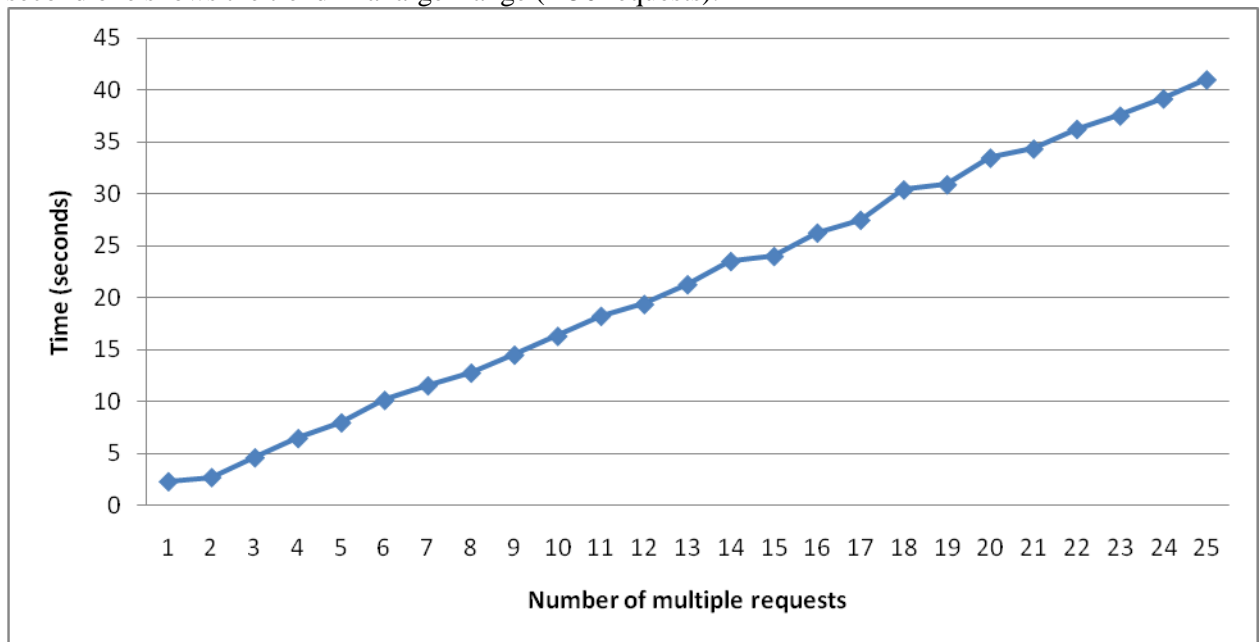


Figure 13 - Average response time of path finding workflow (1-25 requests)

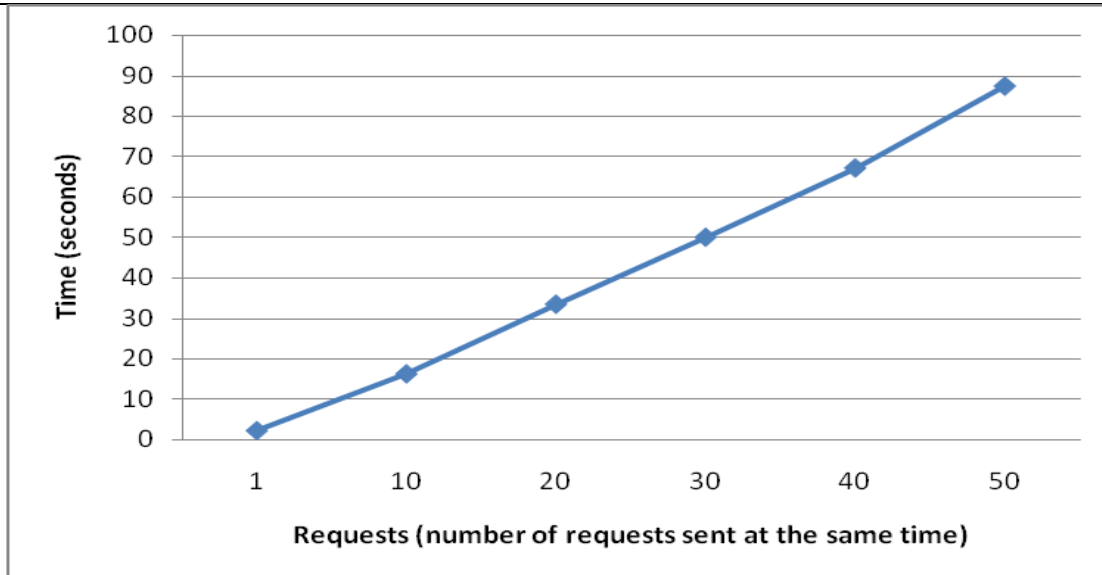


Figure 14 – Average response time of path finding workflows

4.5.6. Considerations

Observing the results we presented in the previous section we can state that the response time increases when the number of requests grows. As it's possible to see in the two graphical representations in Figure 13 and in Figure 14, the response time trend can be described by a linear function with y-intercept around on zero and gradient about on 1.65:

$$\text{Average response time} = 1.65 * \text{Number of requests}$$

This isn't a good result: it means that if the number of requests doubles, the average response time doubles too and the average response time requires about 1.65 second more for each additional concurrent request.

The main reason for this behaviour is given by the fact that every workflow is executed independently from the others: as we explained in Section 4.5.2 this workflow is the one that mostly depends by the processor of the machine where LarKC is executed; it means that every time it receives a requests LarKC processes the RDF graph of Milano converting it in the suitable format and applies Dijkstra algorithm to it.

4.6. Alpha Urban LarKC stress test – Monument workflow

4.6.1. Test goals

This test shares the objectives of the previous stress test described in Section 4.5.1: collect information about the Alpha Urban LarKC related to a growing number requests sent at the same time in order to obtain information about the application and LarKC.

The main difference from the previous test is the environment: in fact in this case the application doesn't interact only with elements located in a LAN but it will operate with services located in the Web. As we will see below, this fact influences the results of the test. We think that it's important to test also LarKC in this way, to understand the behaviour of LarKC in more complex context.

4.6.2. Considered workflow

As showed in Figure 15 the monuments workflow is composed by four plug-ins: a Transformer that processes the input query, an Identifier that looks for the data connecting to Sindice and DBpedia, a Selector that prepare the RDF graph to be processed by the last component, and the Reasoner (more information is available on D6.5).

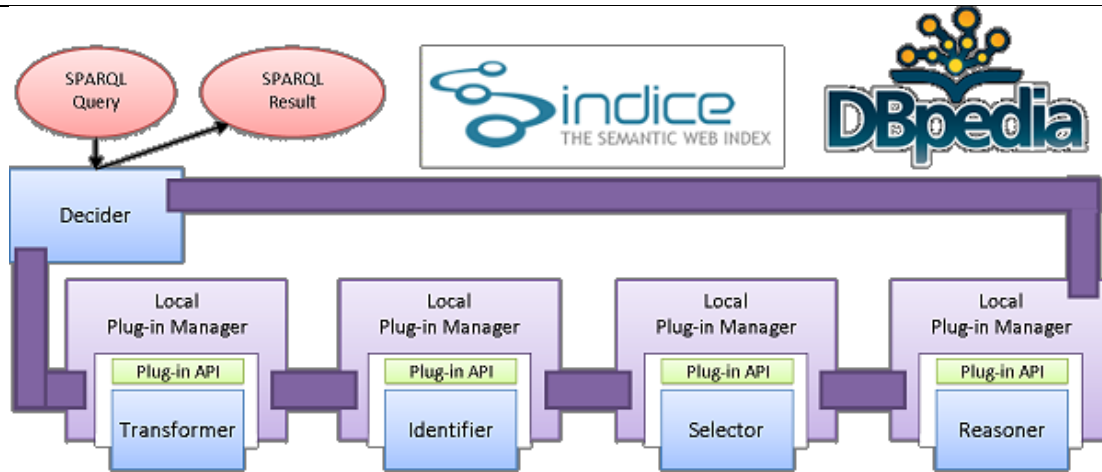


Figure 15 - Monument workflow

We collected some information about the execution time of each plug-in in this workflow and with reference to results shown in Table 8 we can observe that the Identifier plug-in is the one that requires more time to be executed. In fact the main operation performed by this workflow is the retrieval (using Sindice) and the download of data related to Milano monuments (from DBpedia) and it's the task executed by the Identifier, while the Reasoner plug-in has to perform a SPARQL select query to the data collected by previous plug-ins.

| Plug-in | Execution percentage | Execution time |
|-------------|----------------------|----------------|
| | | seconds |
| Transformer | 0.01% | 0.0004 |
| Identifier | 90.69% | 3.5006 |
| Selector | 0.04% | 0.0016 |
| Reasoner | 9.26% | 0.3574 |

Table 8 - Execution time allocation to monument workflow plug-ins

This short analysis is important to understand that the main LarKC operation in the execution of this workflow is the interaction with the Web to collect the required data.

4.6.3. Methodology

The methodology is the same of the path finding workflow stress test. A description can be found in Section 4.5.3.

4.6.4. Environment

In Figure 16 the environment we used in the stress test with the monuments workflow is showed.

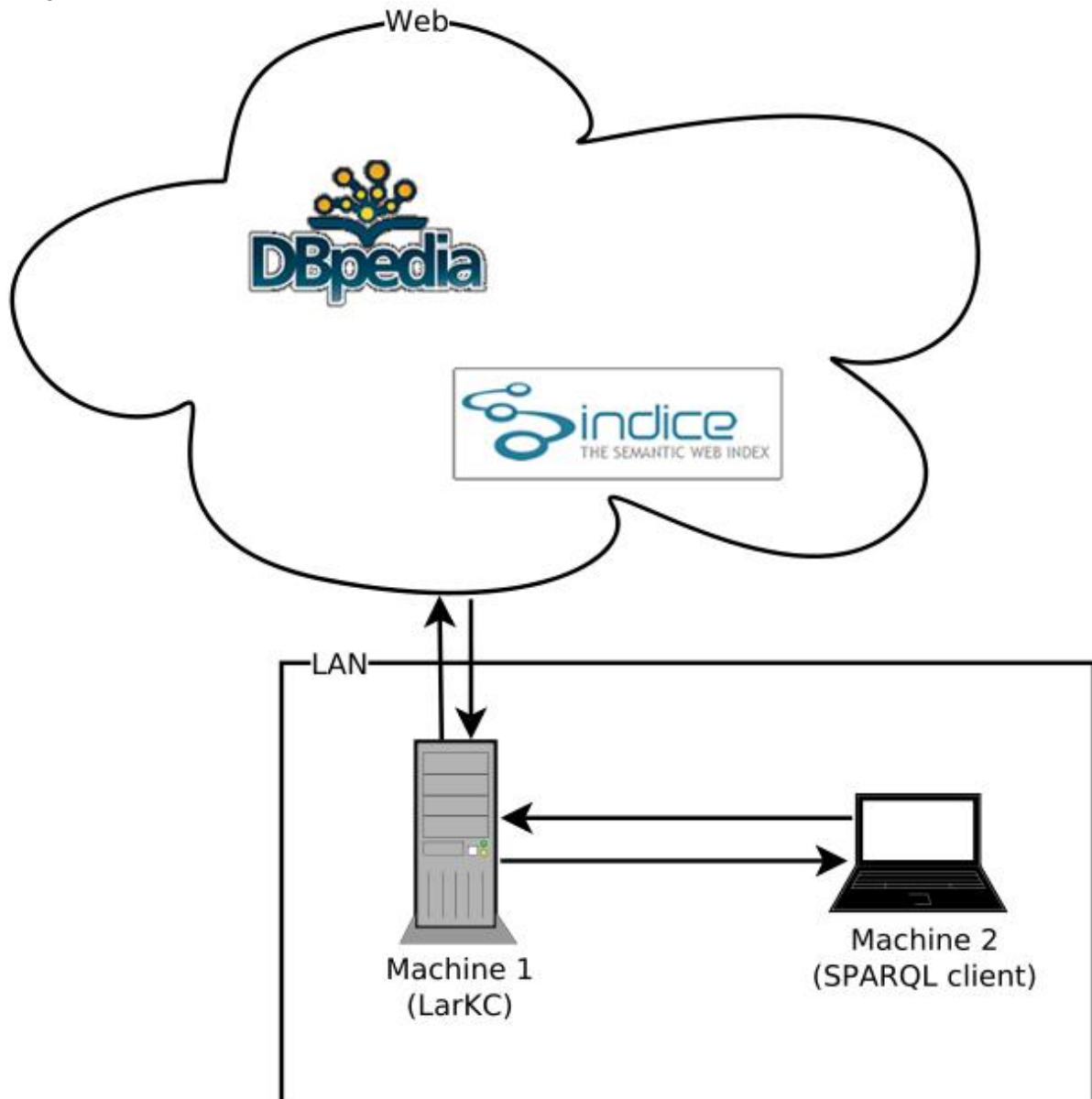


Figure 16 - Enviroment used in the monuments stress test

The machines we used are the same described in Section 4.5.4; Machine 1 (where LarKC was executed) has the following configuration:

- Processor: Intel Xeon 3.60/3.60 GHz (two dual core CPUs)
- RAM: 4 GB
- Operating system: Microsoft Windows Server 2003 Standard Edition 32 bit (Service Pack 2)
- LarKC:
 - o Revision: 744 (August 25, 2009 2:35 PM)
 - o JVM Heap Memory: 512 MB

Machine 2 (used to send requests to LarKC) had the following configuration:

- Processor: Intel Core 2 2.16 GHz
- RAM: 2 GB
- Operating system: Microsoft Windows XP Professional (Service Pack 3)



4.6.5. Test results

In this test we collected data regarding multiple requests whole the number of requests goes from 1 to 25 (see Table 7). A graphical representation of the collected values is also available in Figure 17.

| Number of requests | Mean | Variance | Number of requests | Mean | Variance |
|--------------------|----------|----------------------|--------------------|----------|----------------------|
| | Seconds | seconds ² | | seconds | seconds ² |
| 1 | 7.731385 | 0.129214 | 14 | 7.281642 | 0.553042 |
| 2 | 7.217187 | 0.298958 | 15 | 7.073479 | 0.412275 |
| 3 | 6.688632 | 0.230971 | 16 | 7.020059 | 0.204132 |
| 4 | 7.226892 | 0.207342 | 17 | 7.216723 | 0.27665 |
| 5 | 6.43811 | 0.015626 | 18 | 7.553702 | 0.607905 |
| 6 | 6.866993 | 0.14952 | 19 | 7.855471 | 1.274345 |
| 7 | 6.842114 | 0.323052 | 20 | 7.496533 | 0.143219 |
| 8 | 6.773499 | 0.176878 | 21 | 7.882514 | 0.97397 |
| 9 | 6.796035 | 0.399256 | 22 | 7.56966 | 1.226653 |
| 10 | 6.946855 | 0.593651 | 23 | 7.551817 | 1.098266 |
| 11 | 7.33806 | 1.091308 | 24 | 7.405214 | 0.71124 |
| 12 | 7.253528 | 1.301223 | 25 | 7.710337 | 0.335899 |
| 13 | 7.106755 | 0.34971 | | | |

Table 9 - Results of 1-25 multiple requests with events workflow

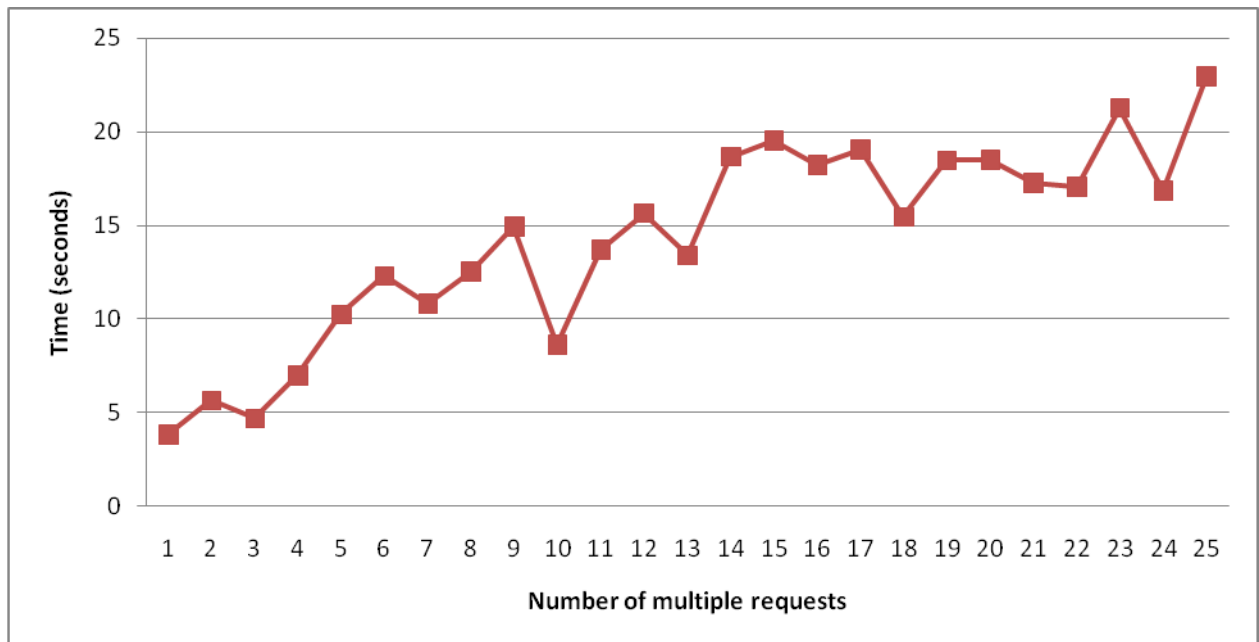


Figure 17 - Average response time of monuments workflow (1-25 requests)



In addition, we also observed that when the number of requests increases LarKC could get some errors, as reported in Figure 18.

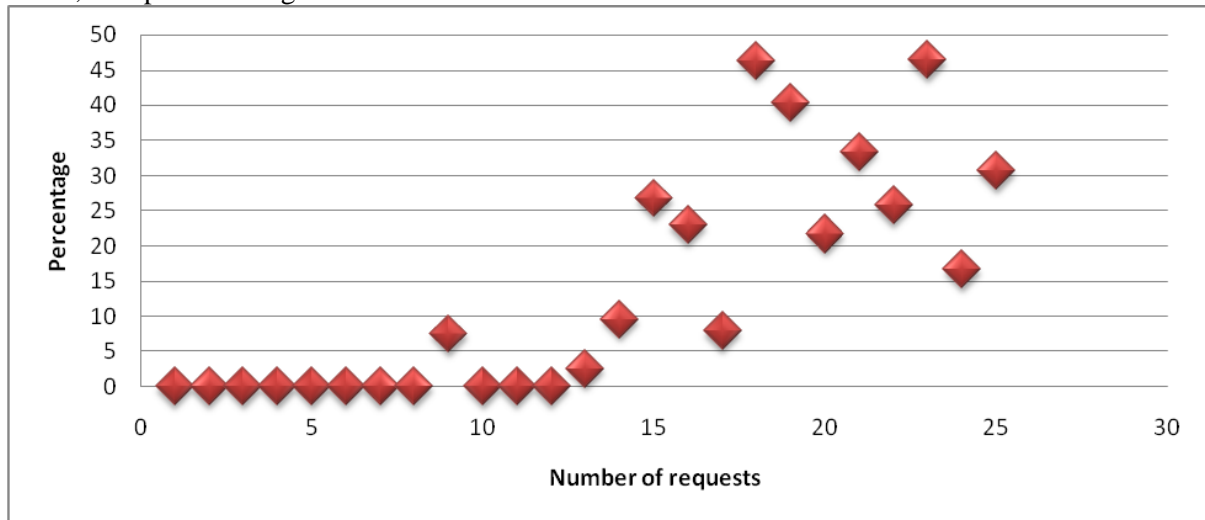


Figure 18 - Errors raised while executing the monument workflow

The x-axis represents the number of requests sent at the same time, while in the y-axis the percentage of errors is showed. We can observe that when more than 10-15 requests are sent there is always at least one error, with peaks of more than 40%. We observed that when error occurs, only a subset of the sent requests receive a response (containing the correct response) from LarKC, while other requests didn't receive any message from the application. Due to this fact, we choose to don't increase anymore the number of requests.

4.6.6. Considerations

This workflow is the one that provided the strangest behaviour: the average response time varied a lot (we can deduce it from the variance values of the average times of the experiments showed in Table 9). The cause of this behaviour are related to the environment: as explained in Section 4.6.2, the main operation of the monument workflow is an interaction with some Web services in order to obtain the location of the searched data (Sindice) and download it (DBpedia). There exists a high dependence upon the external environment, so delays and network problems are introduced, influencing a lot the results we obtained.

Regarding the error issue we reported above, we investigated and we found the main problem: each request is managed independently from the others, so the application tries to get the (same) data from Sindice and DBpedia more than one time. When the number of requests is high some connection could be refused (mainly by DBpedia), the Identifier plug-in is unable of download contents, the workflow execution is blocked and no error is raised. It means that some workflows are executed successfully and some responses are sent back to the clients, but other requests are not satisfied and in these cases no output is sent by LarKC.

4.7. Alpha Urban LarKC stress test – Event workflow

4.7.1. Test goals

The goals of this test are the same of the previous one. See Section 4.6.1 to get information about them.

4.7.2. Considered workflow

The events workflow has been developed to find the events (actually the ones located in Milano) available in Eventful archive. Figure 19 represents the workflow: an Identifier connects to the Eventful REST service in order to collect Milano event links; a Transformer collects the data (formatted in RDF/XML) accessing the links found by the identifier and applies an XSL transformation to convert the format in RDF/XML, then a Selector prepares the data set and the Reasoner executes a SPARQL query to extract the information required by the input query.

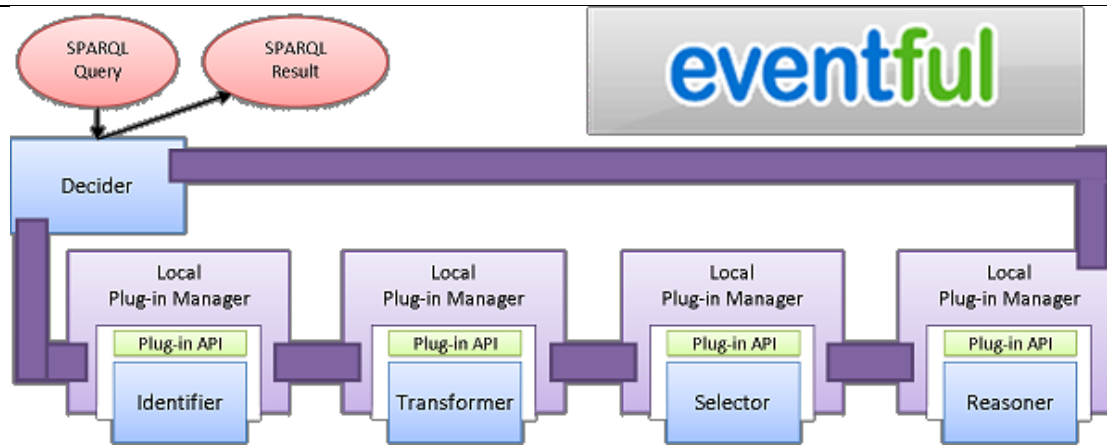


Figure 19 - Events workflow

As in the previous two cases we analyzed the workflow execution to determine how much each plug-in influences the workflow execution; results are presented in Table 10.

| Plug-in | Execution percentage | Execution time |
|-------------|----------------------|----------------|
| | | seconds |
| Identifier | 16.86% | 1.3032 |
| Transformer | 76.26% | 5.8949 |
| Selector | 0.01% | 0.0008 |
| Reasoner | 6.87% | 0.5311 |

Table 10 - Execution time allocation of events workflow plug-ins

Most of the execution time of this workflow is used by the Transformer plug-in: it has two operations: it downloads the event data from Eventful and it performs the XSL transformation to prepare the RDF graphs for the next plug-ins (the first operation requires about 40% of the time spent by the transformer and the remaining ones require the 60%). It means that this workflow is an intermediate case between the previously described workflows: it interacts with Web sources and it stresses the processors with XSL transformations and SPARQL select execution.

4.7.3. Methodology

A description of the methodology we followed in this test can be found in Section 4.5.3.

4.7.4. Environment

The environment of this test is the same of the monument workflow; the only difference is the Web service considered: while in the monument workflow LarKC interacted with Sindice and DBpedia, in this case the interaction is between LarKC and Eventful. To get information about the machine configurations see Section 4.6.4.



4.7.5. Test results

In a similar way to other stress tests we started to execute this test considering a number of requests going from 1 to 25.

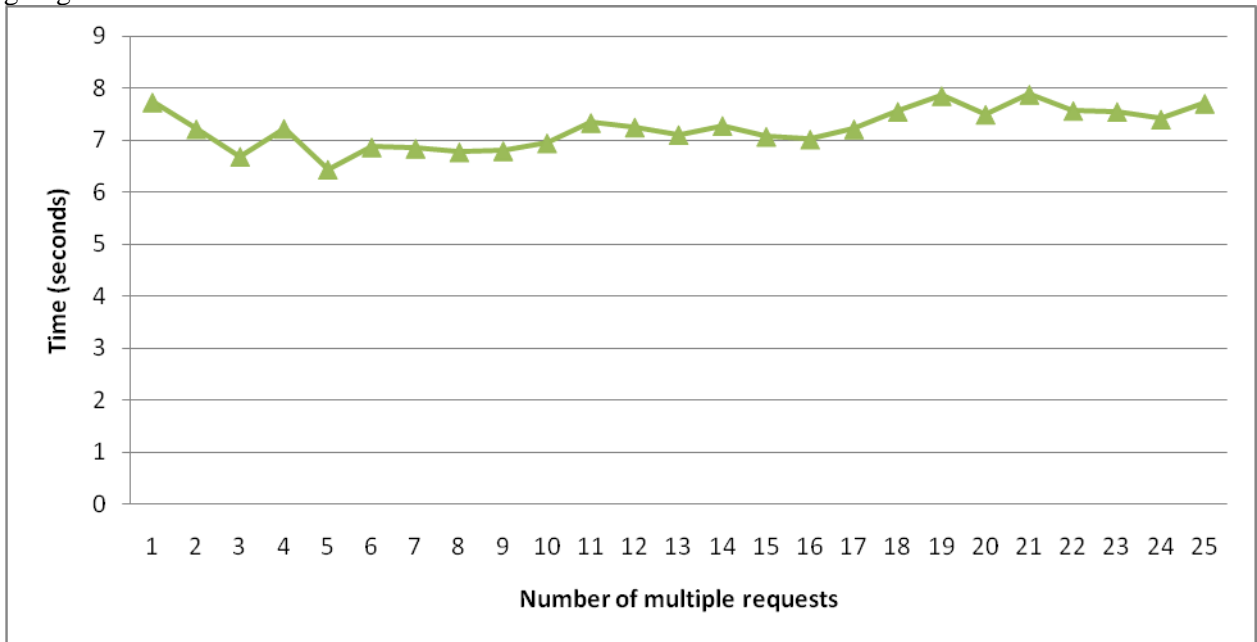


Figure 20 - Average response time of events workflow (1-25 requests)

Observing that the average response time didn't grow significantly (it seemed to remain constant as it's possible to observe in Figure 20) we continued to increase the number of requests sent at the same time to 50 (Figure 21).

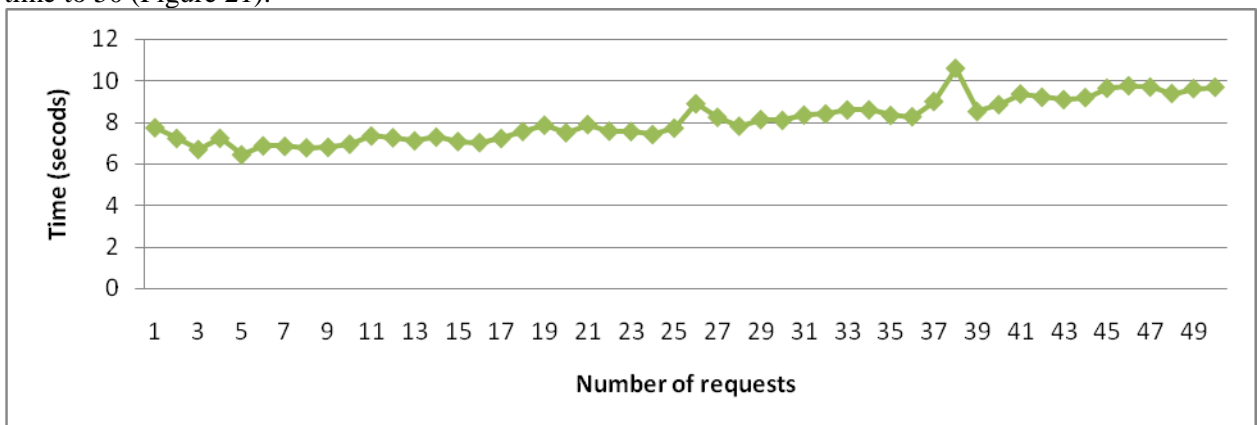


Figure 21 - Average response time of events workflow (1-50 requests)

Then we continued to increase the number of requests up to 200 (every time adding 10 new queries) and collecting a large amount of data.

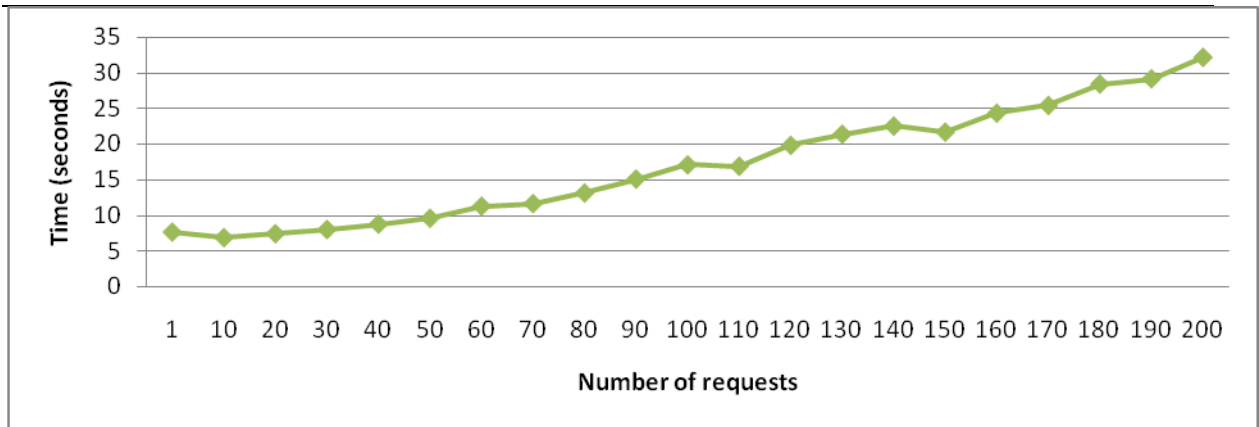


Figure 22 - Average response time of events workflow (1-200 requests)

The data collected and showed in graphs presented above is available in Table 11.

| Number of requests | Mean | Variance | Number of requests | Mean | Variance | Number of requests | Mean | Variance |
|--------------------|----------|----------------------|--------------------|----------|----------------------|--------------------|----------|----------------------|
| | seconds | seconds ² | | seconds | seconds ² | | seconds | seconds ² |
| 1 | 7.731385 | 0.129214 | 23 | 7.551817 | 1.098266 | 45 | 9.62319 | 0.062054 |
| 2 | 7.217187 | 0.298958 | 24 | 7.405214 | 0.71124 | 46 | 9.736557 | 0.04226 |
| 3 | 6.688632 | 0.230971 | 25 | 7.710337 | 0.335899 | 47 | 9.676861 | 0.673182 |
| 4 | 7.226892 | 0.207342 | 26 | 8.876962 | 0.102735 | 48 | 9.366142 | 0.136265 |
| 5 | 6.43811 | 0.015626 | 27 | 8.231625 | 0.044616 | 49 | 9.601006 | 0.825055 |
| 6 | 6.866993 | 0.14952 | 28 | 7.80746 | 0.967046 | 50 | 9.667205 | 0.106868 |
| 7 | 6.842114 | 0.323052 | 29 | 8.1201 | 0.294417 | 60 | 11.35537 | 0.068661 |
| 8 | 6.773499 | 0.176878 | 30 | 8.084858 | 0.289689 | 70 | 11.7154 | 0.081767 |
| 9 | 6.796035 | 0.399256 | 31 | 8.340133 | 0.179199 | 80 | 13.19959 | 0.120149 |
| 10 | 6.946855 | 0.593651 | 32 | 8.391032 | 0.378973 | 90 | 15.13896 | 0.238838 |
| 11 | 7.33806 | 1.091308 | 33 | 8.581132 | 0.019094 | 100 | 17.1286 | 0.318297 |
| 12 | 7.253528 | 1.301223 | 34 | 8.586725 | 0.360113 | 110 | 16.93789 | 4.716742 |
| 13 | 7.106755 | 0.34971 | 35 | 8.31986 | 0.247036 | 120 | 19.84971 | 0.220206 |
| 14 | 7.281642 | 0.553042 | 36 | 8.254429 | 0.079587 | 130 | 21.41304 | 0.251049 |
| 15 | 7.073479 | 0.412275 | 37 | 8.985154 | 0.15674 | 140 | 22.54884 | 0.659482 |
| 16 | 7.020059 | 0.204132 | 38 | 10.5694 | 1.819463 | 150 | 21.7099 | 0.277549 |
| 17 | 7.216723 | 0.27665 | 39 | 8.504194 | 0.138683 | 160 | 24.35767 | 3.577357 |
| 18 | 7.553702 | 0.607905 | 40 | 8.837178 | 0.176556 | 170 | 25.43722 | 5.561891 |
| 19 | 7.855471 | 1.274335 | 41 | 9.34349 | 0.120763 | 180 | 28.43282 | 0.085401 |
| 20 | 7.496533 | 0.143219 | 42 | 9.18503 | 0.081149 | 190 | 29.16314 | 4.498249 |
| 21 | 7.882514 | 0.97397 | 43 | 9.076416 | 0.021306 | 200 | 32.1658 | 0.471285 |
| 22 | 7.56966 | 1.226653 | 44 | 9.172601 | 0.832636 | | | |

Table 11 - Collected data about the events workflow performance

As in the monuments workflow, also in this case we encountered some problems: when the number of requests was higher than 100, some connections with LarKC platform failed (less than 10%), as it's possible to observe it in Figure 23.

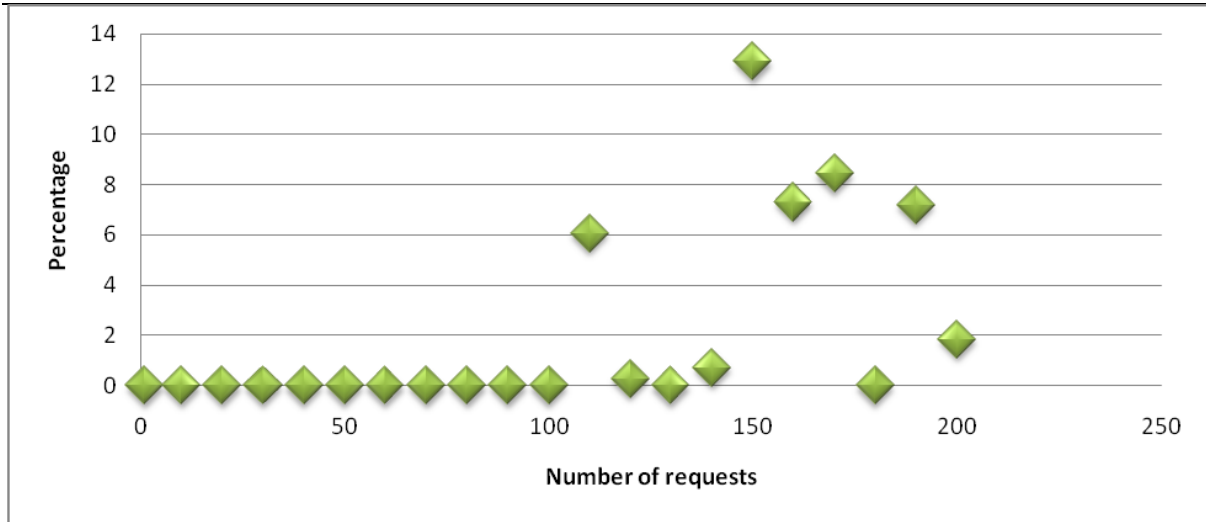


Figure 23 - Errors raised while executing the events workflow

4.7.6. Considerations

The behaviour of this pipeline is similar to the one observed in other stress tests: the average response time increases following a linear function. Analyzing the collected data we can approximate the function describing the average response time:

$$\text{Average response time} = 0.121 * \text{Number of requests sent at the same time} + 5$$

This result is different from the one obtained in the path finding stress test (Section 4.5.6): in fact this case we have a slope very low and a y-intercept around on 5 seconds. Every new requests received by LarKC requires around 0.121 seconds to be satisfied, it is around 13 times lower than the time required by the path finding workflow to response to an additional request.

Regarding the errors we found in the test, actually we didn't find if the cause is LarKC or the program we wrote to make the test, we will investigate in next months about the problem causes.

4.8. Rule-based path finding stress test

4.8.1. Test goals

The goal of this test is to analyze the behaviour of the described pipeline while increasing the number of concurrent user requests.

4.8.2. Considered workflow

The considered workflow is the same described in Section 4.4.1 and graphically represented in Figure 10 on page 22.

4.8.3. Methodology

In a similar way to the tests described in Section 4.4, we focused our attention on “comparison test by user increase”. The goal of this test is to verify how LarKC performance measurements are influenced by the number of concurrency requests required to find the path and comparing them to the RB-ToC ones.

4.8.4. Environment

The environment for this test is the same described in Section 4.4.3 on page 23.



4.8.5. Test results

The following is to test the Path pipeline with the OntoBroker reasoner and RB-ToC with the increase of concurrent users. Figure 24 shows the elapsed mean time.

| Number of concurrent requests | Total workflow execution (sec) | Identifier plug-in execution (sec) | Reasoner plug-in execution (sec) | Rule-based ToC execution (sec) |
|-------------------------------|--------------------------------|------------------------------------|----------------------------------|--------------------------------|
| 1 | 66.17 | 65.72 | 0.41 | 0.77 |
| 2 | 103.20 | 102.55 | 0.41 | 0.80 |
| 3 | 82.72 | 81.28 | 0.41 | 1.05 |
| 4 | 106.62 | 106.24 | 0.42 | 1.26 |
| 5 | 109.28 | 108.95 | 0.41 | 1.40 |

Figure 24 - Results of Simultaneous execution using the OntoBroker Reasoner pipeline and RB-ToC

4.8.6. Considerations

The interpretation of this test results is very similar to the consideration expressed in Section 4.4.5. the positive side of this result is that the behaviour of the Reasoner plug-in seems to be almost unaffected by the increase of concurrent requests number.

4.9. Alpha Urban LarKC – Monument workflow – data quality test

In Sections 4.9 and 4.10, we include a final kind of tests that we named “data quality test”. This cannot be properly considered as a “performance” test (as the deliverable title could suggest), since it does not relate with the Urban LarKC performance. Nonetheless, we believe that this evaluation is highly relevant for the Urban Computing use case.

4.9.1. Test goals

The last kind of test that we performed is focused on the quality of the results provided by the Alpha Urban LarKC. We manually checked the data sources involved by our application for the monuments retrieval in order to check how many monuments are available and how many of them are returned by the alpha Urban LarKC. In addition we took note of problems that affect the data (duplicate information, wrong coordinates and so on) in order to try to cope with some of these problems and obtain better results in the next releases of Alpha Urban LarKC. This and other considerations are described in more detail in Chapter 5.

The following Section 4.10 is devoted to the same kind of test, but in relation with the events retrieved by the event workflow.

4.9.2. Considered workflow

The considered workflow is described in Section 4.6.2.

4.9.3. Methodology

In this test with the monument pipeline we consider the DBpedia source, which is quite a “static” source, in that the data change rarely, especially compared to the expected requests rate.

We manually compare the results obtained through the invocation of the monument workflow with the data actually contained in the DBpedia source.

4.9.4. Test results

Table 12 shows the results of our surveys on geo information available on Milano monuments on DBpedia. Actually only 6 monuments of 25 available ones have valid coordinates associated with them.

In addition, the Alpha Urban LarKC connects to DBpedia using Sindice service, and we observed that the query submitted to the search engines returns only two results (of six that should be found). This means that Sindice doesn’t successfully index the full content of the DBpedia RDF files.



| | |
|---|----|
| Available monuments | 25 |
| Monuments without coordinates | 17 |
| Monuments with W3 geo coordinates | 6 |
| Monuments with DBpedia properties | 0 |
| Monuments with coordinates in a linked node | 3 |
| Monuments with wrong coordinates in a linked node | 3 |
| Monuments indexed by Sindice with valid coordinates | 2 |

Table 12 - Available geo information of monuments in Milano (type of coordinates explained below)

4.9.5. Considerations

As explained before, DBpedia contains “static” data because it changes rarely. This change however doesn’t happen so infrequently. Since we started developing LarKC, a new version of DBpedia has been released and it greatly influenced the results of our application presented in the table above. In fact, in the latest version of DBpedia, the way to represent geographic information changed. Now it’s possible to find those geographic RDF triples in three different formats:

- *W3C geo vocabulary* (this format was already used in previous versions of DBpedia): latitude and longitude are expressed in decimal format using geo:lat and geo:long properties (geo is the prefix of the namespace: http://www.w3.org/2003/01/geo/wgs84_pos#). An example of a resource using this format is available here: http://dbpedia.org/page/Castello_Sforzesco;
- *DBpedia properties*: a set of properties defined by BDpedia (such as dbpprop:latd, dbpprop:latm, dbpprop:latns, etc.) has been introduced to describe the coordinates expressed in DMS format. At this link: <http://dbpedia.org/page/Milan> it is possible to find an example of resources annotated in this way;
- *Geographic node*: a DBpedia-defined property (dbprop:geo or dbprop:latLong) links the resource with another resource representing its location. For example, the resource describing Paris (<http://dbpedia.org/page/Paris>) has a dbprop:latLong property with range in the resource: <http://dbpedia.org/resource/Paris/latLong/coord>, that contains the coordinates of Paris.

The employment of these new additional formats seems to have introduced some problems in DBpedia, since some monuments, that in the previous DBpedia version had correct coordinates, lost their geographic information. For example, http://dbpedia.org/page/Milan_Cathedral, representing the Duomo of Milano, is related to http://dbpedia.org/resource/Milan_Cathedral/geo/coord with the relation <http://dbpedia.org/property/geo>, but the resource that should contain the coordinates doesn’t contain them.

As a consequence, the query we use in our Urban LarKC pipeline, which makes use of the W3C geo vocabulary, can get only a subset of the available information.

4.10. Alpha Urban LarKC – Event workflow – data quality test

4.10.1. Test goals

Similarly to the test described in the previous section, this test aims to evaluate the quality of the data retrieved by using the event workflow. We manually checked the data sources involved by our application for the events retrieval in order to check how many events are available and how many of them are returned by the alpha Urban LarKC.

4.10.2. Considered workflow

The considered workflow is described in Section 4.7.2.

4.10.3. Methodology

In this test, we consider the Eventful source for the events data. While in the previous section DBpedia was considered a “static” source (the data changes rarely), Eventful contains “dynamic” data (there is a continuous addition of events). Due to this fact, we choose to collect Eventful data for a time interval (a week) and grouped them by day.



4.10.4. Test results

Results retrieved in the week between September 5th and September 11th are available in Table 13: every day about 10-15 events related to Milano are published and some of them have been inserted more than once. The table shows that the data has some problems: we checked the coordinates of the venues, the date and if the location is different from the city of Milano in Italy.

| | Total | Unique | Problems | | |
|-----------|-------|--------|--------------|------------|---------------|
| | | | Wrong coords | Wrong date | Not in Milano |
| 9/5/2009 | 12 | 11 | 9 | 1 | 1 |
| 9/6/2009 | 12 | 11 | 9 | 1 | 1 |
| 9/7/2009 | 13 | 10 | 8 | 1 | 1 |
| 9/8/2009 | 14 | 12 | 11 | 1 | 1 |
| 9/9/2009 | 19 | 15 | 11 | 1 | 1 |
| 9/10/2009 | 14 | 11 | 7 | 1 | 0 |
| 9/11/2009 | 11 | 9 | 7 | 1 | 2 |

Table 13 - Events retrieved in Eventful

4.10.5. Considerations

Going more deeply in the problems we found in this data, the main ones are:

- Missing coordinates: it happens that coordinates of venues are missing; when it happens Eventful assigns a default location to the venue where the event is planned (a point near the centre of Milano);
- Wrong coordinates: some venues are associated with the wrong coordinates; for example we found events where the venue was San Siro stadium or FieraMilano, but the coordinates pointed in other points (at worst other cities);
- Wrong places: some events are associated to Milano but they are in nearby cities (like Como or Verona, approximately 50 and 150 kilometres respectively from Milano);
- Wrong “Milano”: when we search events with the city set as “Milan” we could also find events located in other places with the same name (for example there is a Milan near Nashville, Tennessee);
- Wrong date: some events are associated with wrong dates; in fact in the event description there is text with information about the event and it contains the correct date, while in the event start time property another (different) date is reported.

A deeper discussion on those points is offered in Chapter 5.

4.11. Traffic Prognosis

Since the main dataset for traffic forecasting became available in July 2009, we cannot yet report results for quality and performance measures. Instead, we sketch a number of measures suitable for determining the prediction accuracy as well as the run time of our planned multi-stage approach; for a detailed description of the approaches, see D6.5, Section 4.

Specifically, we will measure the computation time of our neural network models, which are explained in D6.5, with respect to the training and recall phase. During the training the neural network learns from data, whereas during the recall an input pattern is presented to the network and the output is computed. To the test timings, we will add the time to smooth the predicted speed values at the sensor locations onto the whole street grid. We will measure the computational real time, which is defined as the elapsed time from beginning to end of training, recall respectively.

The forecast accuracy of our learning-based models is evaluated with standard and problem specific measures. Concerning standard performance measures, we will refer to the mean square error (MSE) and to mean absolute percentage error (MAPE). In addition to these (double-sided) measures, we propose to evaluate the forecast accuracy of the models with left- or right-sided measures that directly refer to the underlying traffic control policy resp. problem setting. For instance, to optimize the traffic throughput it could be more favourable



to overestimate traffic volume than to underestimate it. All performance measures are calculated separately on in- and out-of-sample data for the forecast horizons of interest.



5. Lesson learned and challenges for LarKC

The test results obtained confirmed some of the problems we raised in D6.3 Section 3. In fact, in this case we acted as end-users of the application and we encountered some problems.

The first issue, that should be dealt with to avoid undesirable situations, is the *exception handling*: when a plug-in gets an error and raises an exception (not handled by it), the workflow stops and no response is sent to the requester. We think that an error notification mechanism is very important and should be inserted in the LarKC platform, because it's not possible to assume that every plug-in will be designed and developed avoiding exceptions-raising. For example, returning an HTTP 500 message (internal server error) is better than don't send any message to the (waiting) client.

The second issue is related to the stress tests results: the *growth of the response time* when the number of requests increase is linear. Currently, the concurrency is managed in a trivial way: every new query is managed separately from the others (in a new thread). There are some situations that are not considered in this approach, for example:

- A similar query has just been answered, so the requested data is in the Data Layer and it's possible to access it without retrieving it again from external sources;
- LarKC is solving a query the same as one that just arrives, so the answer could be shared (without processing it again); this happens when the change rate of the data is much slower than the user requests' rate (e.g. the monuments in Milano don't change so often to require retrieving their list for each user request).
- A plug-in has to perform an operation that requires a large amount of resources (for example time or processor) and its result can be used for different queries, but it is executed in every workflow.

Some of these problems could be addressed in plug-ins development, but the more the platform manages these situations properly, the easier and faster will be the design and the development of plug-ins. For example, the LarKC Data Layer supports persistence of data, but it's not clear how to manage it (should it be controlled by the platform? or should it be the Decider plug-in who manages it?). A *cache system* or something like that could be very useful.

Another consideration regards the rule-based path finding workflows: the test we performed with OntoBroker and its wrapper plug-in within a LarKC platform was designed in order to test LarKC in its current version and structure. Therefore, the path finding workflow described in Section 4.4.1 loads the data immediately before computing the query. However, a Reasoner like OntoBroker is not intended for in-memory loading and processing of data, since its main value shows up when data is loaded in advance and the reasoning materialization already took place.

Our suggestion and feedback to the LarKC platform developers is to define new workflow construction modalities that allow for a *"batch-time" workflow* to anticipate data loading to allow for the reasoning materialization of OntoBroker and a *"run-time" workflow* to process and answer the actual path-finding query. This would allow for a better exploitation of OntoBroker capabilities.

A lesson learned that could seem quite obvious regards the *use and invocation of third-party services*, which are not necessarily trustworthy. As described in Sections 4.9 and 4.10, we experienced some problems in terms of both functionalities and quality of results when using external services like Sindice and DBpedia. In order to overcome those problems, we could think of bypassing them, for example by using a LarKC-developed service like Linked Data Semantic Repository (LSDR, available at <http://ldsr.ontotext.com/>), which should include the same data about Milano monuments.

Finally, a general lesson learned of the first period of the LarKC project is that the pluggable platform the project is developing is *trading performances for flexibility*: the platform results very extensible, but this can hinder performances by causing a computational overhead with regards to comparable systems with less flexibility. This overall remark, which applies to the whole project results, became evident also in our Urban Computing use case tests. As reported in Sections 4.1-4.4, when comparing the Urban LarKC application with their respective Terms of Comparison, a manifest overhead in terms of response time was recorded.

Besides the lesson learned and suggestions for improvements described so far, our early evaluation highlights a number of challenges that could be interesting for the technical work-packages in LarKC.



The first challenge derives from the tests reported in Section 4.10 about the results of the search for events in Milano with Eventful. We experienced a problem of ***duplicated information*** (e.g. the same event reported more than once), a common problem that can show up in Web 2.0-like services when different users collaboratively contributing to a system provide the same information in slightly different formats. The challenge for LarKC could be to try applying some rule-based approaches or machine-learning technologies to solve this problem (e.g. same place, same time and similar description mean same event or training on manually-detected duplicates).

Another challenge derives from the tests reported in the same section referenced above. We experienced also a problem of wrong ***disambiguation of geographical names and identifiers***, since a query to Eventful for “Milano” returned data about the Italian city as well as information related to other places around the world with the same name of “Milano”. The challenge for LarKC could be to try applying some reasoning approach to solve this issue. A possible step towards the solution is offered in [1].

The last challenge which is intrinsic to the project objective is the ability to deal with ***very-large scale data***, such as the traffic sensor data about Milano described in Section 3.1. As explained in this document as well as in the companion deliverable D6.5, the traffic prediction techniques will be the first ones to address this challenge with the data size; we do not exclude the future possibility for other LarKC explored technologies to deal with this issue as well.



6. Conclusions

In this deliverable we updated the list of data sources (continuing what we started in D6.2) that we analyzed in previous months and we'd like to integrate in our scenario in the near future. These data sources contain traffic-related information (traffic history, information about weather and so on) and user-generated stream data, useful for analyzing how to integrate this kind of data into our urban scenario.

On the other side we started to conduct tests in order to evaluate the performance of various LarKC configurations, using what we developed in past months: some path finder workflows working with different policies and approaches (for example using operational research approach and rule-based one) and the Alpha Urban LarKC, one of the first applications developed using LarKC.

A summary of the test results and their value in terms of lesson learned and possible challenges for the LarKC project was offered in Chapter 5.

We intend to repeat the tests we performed in this deliverable in the next data and performance ones (D6.7, D6.9 and D6.11), adding new kinds of tests if necessary, in order to compare the indicators we obtained in this deliverable with the following ones. In order to get better results, we will focus on the points contained in Chapter 2 and we will try to continue our activities described in D6.5.

7. References

- [1] Raphael Volz, Joachim Kleb, Wolfgang Mueller: "*Towards ontology-based disambiguation of geographical identifiers*", in Proceedings of the WWW2007 Workshop i³: Identity, Identifiers, Identification on Entity-Centric Approaches to Information and Knowledge Management on the Web, co-located with WWW2007, Banff, Canada, May 8, 2007.